Computer Methods in Applied Mechanics and Engineering 462 (2026) 119295

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Locking-free energy-based physics-informed neural networks for shear-deformable beams and plates

Till Weithoff [a],*, Lukas Striefler [b], Bastian Oesterle [b], Josef Kiendl [a]

[a] University of the Bundeswehr Munich, Institute of Engineering Mechanics and Structural Analysis, Werner-Heisenberg-Weg 39, 85577, Neubiberg, Germany

[b] Hamburg University of Technology, Institute for Structural Analysis, Denickestraße 17, 21073, Hamburg, Germany

ARTICLE INFO

Keywords:
Physics-informed neural networks
Structural mechanics
Shear locking
Hierarchic formulations
Beams
Plates

ABSTRACT

Physics-informed neural networks (PINNs) embed governing equations into the training objective and provide a data-efficient machine-learning framework for problems governed by partial differential equations (PDEs). When applying the PINN framework to stiff PDEs, a deterioration of training speed and accuracy can be observed. From a mathematical perspective, the high ratio of coefficients in a stiff PDE is the source of locking effects, known from other discretization schemes. This work investigates transverse shear locking in PINNs for Timoshenko beams and Reissner–Mindlin plates and proposes effective mitigation strategies. We implement energy-based PINNs using different parametrizations of the kinematics: the standard parametrization and hierarchic parametrizations. Our results demonstrate that standard approaches exhibit severe locking behavior, with training speed deteriorating significantly as the slenderness increases. Based on an in-depth study of the training dynamics, we identify the causes of transverse shear locking in PINNs considered in this work. To address them, we adapt hierarchic formulations from structural mechanics to the PINN framework by: (1) using hierarchic formulations that directly parametrize the shear deformation, (2) employing separate neural networks for each primary variable to isolate the trainable parameters, and (3) introducing a slenderness-dependent scaling factor for the shear deformation variable to balance the gradients of the energy terms for training. The proposed method achieves fast, slenderness-independent training and convergence for both beam and plate problems for all variables of interest. The method is developed using energy-based forward-solving PINNs, and its transfer to related PDE-loss-based learning frameworks is illustrated with a parametric PINN.

1. Introduction

Deep learning has achieved remarkable success in a wide range of applications, including speech recognition, machine translation, and computer vision [1]. This success has motivated its application in scientific computing and engineering, where the solution of partial differential equations (PDEs) is a central task. Classical numerical methods such as the finite element method (FEM) are highly advanced and reliable, but they remain computationally demanding in real-time and multi-query settings, such as optimization and uncertainty quantification. Machine learning methods therefore offer a promising alternative, as they can provide fast evaluations after an initial offline training phase. However, purely data-driven approaches typically require large amounts of training data, which are often unavailable in engineering applications.

* Corresponding author.
E-mail address: till.weithoff@unibw.de (T. Weithoff).

https://doi.org/10.1016/j.cma.2026.119295
Received 2 February 2026; Received in revised form 8 July 2026; Accepted 5 August 2026
Available online 12 August 2026
0045-7825/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

To overcome this limitation, Raissi et al. [2] introduced physics-informed neural networks (PINNs), which incorporate physical laws directly into the training objective. By embedding the governing equations into the loss function, forward-solving PINNs can be trained with little or no observational data while naturally providing additional quantities of interest, such as sensitivities, through automatic differentiation.

The original PINN framework employs a composite loss function consisting of a PDE residual term and boundary condition terms. Although the first PINN formulations were based on the strong form of the PDE, the framework was soon extended to variational formulations, including variational PINNs (VPINNs) [3] and energy-based approaches such as the Deep Ritz Method (DRM) [4] and the Deep Energy Method (DEM) [5]. PINNs have since been applied to a broad range of problems in computational mechanics, including linear elasticity [6–8], hyperelasticity and viscoelasticity [9,10], plates [11,12], and shear-deformable shells [13]. Further applications include contact mechanics [14,15], fracture mechanics [16–18], and topology optimization [19,20].

A central limitation of standard PINNs is their lack of generalization across problem instances, since the network must typically be retrained when the problem setup changes. To address this limitation, several related frameworks have been proposed, including physics-informed deep operator networks [21,22], variational physics-informed neural operators [23], and parametrized physics-informed neural networks (P^2INNs) [24]. Despite their increased generality, these methods still rely on PDE-based loss terms of the same type as standard PINNs. PINNs therefore provide a particularly convenient setting for studying the training behavior induced by PDE-based losses and for developing mitigation strategies that can later be transferred to more general operator-learning frameworks.

Locking is a phenomenon well known from other discretization schemes such as FEM. In this context, it is characterized by an overly stiff response of the numerical model, a reduced convergence rate for coarse discretizations and oscillations of certain stress components or stress resultants. The intensity of all symptoms depends on a so-called critical parameter, e.g., the element slenderness ratio. From a mechanical perspective, locking is usually understood as the inability of an element to display certain deformation states, e.g., pure bending. The source of this issue is mostly associated with imbalanced function spaces used to approximate the solution. Different types of locking phenomena exist for different types of structural models, such as membrane locking, Poisson-thickness locking and transverse shear locking, which are summarized in [25,26]. This paper focuses on transverse shear locking, which for brevity we will refer to as shear locking in the following.

Effective techniques to mitigate locking in FEM implementations have been developed over the last decades, including reduced integration and mixed formulations, among others. While locking is well known in the context of FEM, it is important to be aware that it is not rooted in the finite element method itself, but rather the underlying PDE, which, in a mathematical sense, is a stiff PDE when locking occurs, and can thus also manifest in other numerical methods, such as isogeometric analysis (IGA, [27–29], among others), subdivision surfaces [30], meshfree methods [31–33] and collocation methods [34–37]. In PINNs, symptoms of locking have been observed previously in [13], where the convergence rate and accuracy for a shear-deformable shell reduced considerably when increasing the slenderness. This has been attributed to the increasing difference in magnitude of the loss terms. In detail, locking in PINNs has only very recently been studied in [38], where volumetric locking in nearly incompressible elasticity problems is addressed using a decomposition-based framework that reformulates the PDE into balanced subsystems. In this work, we investigate the occurrence of shear locking in PINNs for shear-deformable beams and plates based on the Timoshenko beam theory and Reissner–Mindlin plate theory, respectively. We implement PINNs based on different formulations known from FEM and IGA, namely the so-called standard and hierarchic formulations [39,40]. The hierarchic formulations are designed to avoid locking on the PDE level by resolving the imbalance of different orders of derivatives of the variables in the shear strains, which is the root cause of shear locking in finite element discretizations.

The remainder of this contribution is organized as follows. First, we introduce the general PINN framework in Section 2. In Section 3, we introduce the Timoshenko beam theory and the Reissner–Mindlin plate theory as well as the respective parametrizations and discuss shear locking. In Section 4, we demonstrate and analyze shear locking in PINNs based on a Timoshenko beam example. In Section 5, we adapt hierarchic parametrizations to the PINN framework using separate networks and a slenderness-based scaling factor. Numerical tests for the proposed PINNs are presented in Section 6 for beams and plates. Lastly, in Section 7, we discuss the results and come to a conclusion.

2. Physics-informed neural networks

Consider a set of PDEs of the general form

$$\mathcal{F}[\boldsymbol{y}(\boldsymbol{x})] = 0, \ \boldsymbol{x} \in \Omega, \tag{1}$$

$$\mathcal{B}[\boldsymbol{y}(\boldsymbol{x})] = 0, \ \boldsymbol{x} \in \partial\Omega, \tag{2}$$

where $\boldsymbol{y}(\boldsymbol{x})$ denotes the solution, $\boldsymbol{x}$ is the spatial coordinate, $\mathcal{F}$ is a differential operator, $\mathcal{B}$ is a boundary condition operator, Ω denotes the domain and $\partial\Omega$ denotes the boundary of the domain. The general PINN setting [2] utilizes deep feedforward neural networks $\mathcal{N}$, which are sometimes called fully-connected neural networks or multilayer perceptrons [1], to approximate the solution $\boldsymbol{y}$ of the PDE. To fit the solution, the network contains trainable weights and biases, which are collected in $\boldsymbol{\theta}$:

$$\boldsymbol{y}(\boldsymbol{x}) \approx \mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta}). \tag{3}$$

To find an optimal set of the parameters $\boldsymbol{\theta}$, a loss function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta})$ is minimized. The original PINN implementation minimizes the loss function

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{L}_{\text{residual}} + \mathcal{L}_{\text{boundary}}, \tag{4}$$

where $\mathcal{L}_{\text{residual}}$ measures the residual of the PDE over the domain Ω, and $\mathcal{L}_{\text{boundary}}$ enforces the boundary conditions on $\partial\Omega$. The gradients of the loss function can be efficiently computed using automatic differentiation [41], a core technique used in machine learning frameworks. PINNs leverage this further by computing the required partial derivatives of the solution w.r.t. the input $\boldsymbol{x}$, i.e., the spatial coordinates, which enables the calculation of the residuals of the loss function.

For certain PDEs, the solution can be found by minimizing a corresponding energy functional over the domain, allowing the replacement of $\mathcal{L}_{\text{residual}}$ with $\mathcal{L}_{\text{energy}}$ [5,19,42]. Using the loss function based on the energy form is advantageous because it reduces the order of derivatives (and as a result, reduces computational cost) required and includes the Neumann boundary conditions. As a consequence, an additional term in the loss function to enforce the Neumann boundary conditions, which might require tuning of an additional weighting parameter, is not necessary. PINNs based on an energy loss have been shown to produce lower errors in less training time in [43], to be advantageous for elastic plates in [11] and more successful when applied to shear-deformable shells in [13] compared to the strong form.

The loss function can further be simplified by directly imposing the Dirichlet boundary conditions on the output of the neural network by multiplying it by a distance function ξ, which satisfies the boundary conditions by design [44]. Thus, the loss function $\mathcal{L}$ from Eq. (4) is reduced to the energy-based loss function $\mathcal{L}_{\text{energy}}$,

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{L}_{\text{energy}}(\boldsymbol{x}, \boldsymbol{\theta}), \tag{5}$$

which is used in the following implementations of PINNs unless stated otherwise.

3. Timoshenko beams and Reissner–Mindlin plates

3.1. Timoshenko beam theory

The Timoshenko beam theory accounts for both bending and shear deformation. The cross-sections of the beam are assumed to remain straight during deformation, but not necessarily perpendicular to the mid-axis. This introduces the shear strain γ to describe the kinematic relations of the beam. With the transverse displacement denoted by w and the total rotation of the cross-section by φ, the kinematic relations are given by the following equation:

$$\gamma = \frac{dw}{dx} + \varphi = w' + \varphi. \tag{6}$$

By defining the bending stiffness as EI and the shear stiffness as GA_S, where E is the Young's modulus, G is the shear modulus, I is the second moment of area, and A_S is the effective shear area, the constitutive equations of the Timoshenko beam are given as

$$M = EI\varphi' = EI\kappa, \tag{7}$$

$$Q = GA_S\gamma. \tag{8}$$

The effective shear area accounts for the assumption of constant shear stress through the thickness. It is smaller than the cross-sectional area A, and is defined via a correction factor k as $A_S = kA$. For rectangular cross-sections, $k = 5/6$ is commonly used.

The total potential energy Π of a Timoshenko beam can be expressed as

$$\Pi = \underbrace{\int_0^L \frac{1}{2} EI\kappa^2 \, dx}_{\text{bending energy } \Pi_{\text{bend}}} + \underbrace{\int_0^L \frac{1}{2} GA_S\gamma^2 \, dx}_{\text{shear energy } \Pi_{\text{shear}}} - \underbrace{\int_0^L fw \, dx}_{\text{external work } \Pi_{\text{ext}}}, \tag{9}$$

where f is the distributed transverse load acting on the beam and L is the length of the beam. In the external work, we neglect the distributed moment, as well as nonhomogeneous Neumann boundary conditions, as those are not relevant for the examples considered in this work.

3.2. Reissner–Mindlin plates

For the following definition of the Reissner–Mindlin plate theory, we make use of Voigt notation to describe the relevant tensors. We also denote the partial derivatives with respect to the spatial coordinates x and y using the notation $()_{,x}$ and $()_{,y}$, respectively.

Similar to the Timoshenko beam theory, the Reissner–Mindlin plate theory accounts for transverse shear deformation by allowing the normal vectors to the mid-surface of the plate to rotate and not remain perpendicular to the mid-surface during deformation. The shear deformation is described by the shear strain vector $\boldsymbol{\gamma}$ given as

$$\boldsymbol{\gamma} = \begin{bmatrix} \gamma_x \\ \gamma_y \end{bmatrix} = \begin{bmatrix} w_{,x} \\ w_{,y} \end{bmatrix} + \begin{bmatrix} \varphi_x \\ \varphi_y \end{bmatrix}, \tag{10}$$

where φ_x, φ_y are the rotation vectors of the normal vector to the mid-surface of the plate and w is the transverse displacement of the mid-surface. The curvature of the plate can be expressed by the derivatives of the rotation tensor:

$$\boldsymbol{\kappa} = \begin{bmatrix} \kappa_{xx} \\ \kappa_{yy} \\ 2\kappa_{xy} \end{bmatrix} = \begin{bmatrix} \varphi_{x,x} \\ \varphi_{y,y} \\ \varphi_{x,y} + \varphi_{y,x} \end{bmatrix}. \tag{11}$$

The constitutive equations for bending and shear in Reissner–Mindlin plates are given as

$$\boldsymbol{M} = \begin{bmatrix} m_{xx} \\ m_{yy} \\ m_{xy} \end{bmatrix} = \boldsymbol{K}_\mathrm{b}\boldsymbol{\kappa} = D \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \kappa_{xx} \\ \kappa_{yy} \\ 2\kappa_{xy} \end{bmatrix}, \tag{12}$$

$$\boldsymbol{Q} = \begin{bmatrix} q_x \\ q_y \end{bmatrix} = \boldsymbol{K}_\mathrm{s}\boldsymbol{\gamma} = kGt \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma_x \\ \gamma_y \end{bmatrix}, \tag{13}$$

where $\boldsymbol{M}$, $\boldsymbol{Q}$ and $\boldsymbol{\kappa}$ represent the components of the bending moment, shear force and curvature, respectively. $\boldsymbol{K}_b$ is the material matrix for bending and $\boldsymbol{K}_s$ is the material matrix for shear, t is the thickness of the plate, ν is the Poisson's ratio and k is again the shear correction factor, given as $k = 5/6$. The bending stiffness D is given as

$$D = \frac{Et^3}{12(1-\nu^2)}. \tag{14}$$

Considering only transverse loading of the plate by f, the potential is given as

$$\Pi = \underbrace{\int_\Omega \frac{1}{2}\boldsymbol{\kappa}^T \boldsymbol{K}_\mathrm{b}\boldsymbol{\kappa}\, d\Omega}_{\text{bending energy}} + \underbrace{\int_\Omega \frac{1}{2}\boldsymbol{\gamma}^T \boldsymbol{K}_\mathrm{s}\boldsymbol{\gamma}\, d\Omega}_{\text{shear energy}} - \underbrace{\int_\Omega w f\, d\Omega}_{\text{external work}}. \tag{15}$$

3.3. Parametrizations for shear deformable beams and plates

In the following, we discuss different parametrizations for shear-deformable beams and plates. The discussion is focused on Timoshenko beams, but the concepts can be directly extended to Reissner–Mindlin plates.

3.3.1. (w, φ)-Parametrization

The (w, φ)-parametrization, often referred to as the standard parametrization, uses the displacement w and the total rotation φ as primary variables. The curvature κ can be computed from φ, and the shear strain γ is computed based on Eq. (6). Rewriting the total potential from Eq. (9) in terms of the primary variables w and φ results in the following potential for Timoshenko beams:

$$\Pi = \int_0^L \frac{1}{2} EI \left(\varphi'\right)^2 dx + \int_0^L \frac{1}{2} GA_\mathrm{S} \left(w' + \varphi\right)^2 dx - \Pi_\mathrm{ext}. \tag{16}$$

The shear stiffness GA_S is proportional to t, while the bending stiffness EI is proportional to t^3. Thus, for slender beams, $GA_\mathrm{S} \gg EI$, which leads to Eq. (16) being a stiff potential. We define the slenderness s for the Timoshenko beam as L/t. Due to the difference in scale of the bending and shear stiffness for slender beams, small errors in the shear strain γ result in large errors in the shear energy. This makes it difficult for numerical methods to accurately approximate the solution. In the finite element context, the requirement $\gamma \approx 0$ for slender beams is often not satisfied by low-order interpolation functions chosen based on the isoparametric concept, due to the imbalanced function spaces of w' and φ in the shear strain $\gamma = w' + \varphi$. This results in an overestimation of the shear energy and thus a stiffening of the system, which is referred to as shear locking.

3.3.2. (w, γ)-Parametrization

The (w, γ)-parametrization [30,45,46] discretizes the displacement w and the shear strain γ as primary variables. The shear strain γ describes the rotation of the cross-section due to shear and can also be referred to as the shear rotation. The total rotation φ is computed based on the kinematic relation in Eq. (6), rearranged to

$$\varphi = \gamma - w'. \tag{17}$$

The potential from Eq. (9) is thus given as

$$\Pi = \int_0^L \frac{1}{2} EI \left(\gamma' - w''\right)^2 dx + \int_0^L \frac{1}{2} GA_\mathrm{S}\gamma^2\, dx - \Pi_\mathrm{ext}. \tag{18}$$

Due to the direct parametrization of the shear strain γ, the requirement $\gamma \approx 0$ for slender beams can be easily accommodated by any discretization scheme, thereby mitigating shear locking. The imbalance of the function spaces is essentially moved from the shear to the bending term of the potential energy which, in theory, causes problems in the thick limit, but, in practice, is not relevant. For classical FEM, this formulation is unsuitable due to the second derivatives required, but in IGA, this formulation can be discretized due to the higher continuity of the basis functions, resulting in a formulation free from shear locking, but mild oscillations in the shear force remain, see [39].

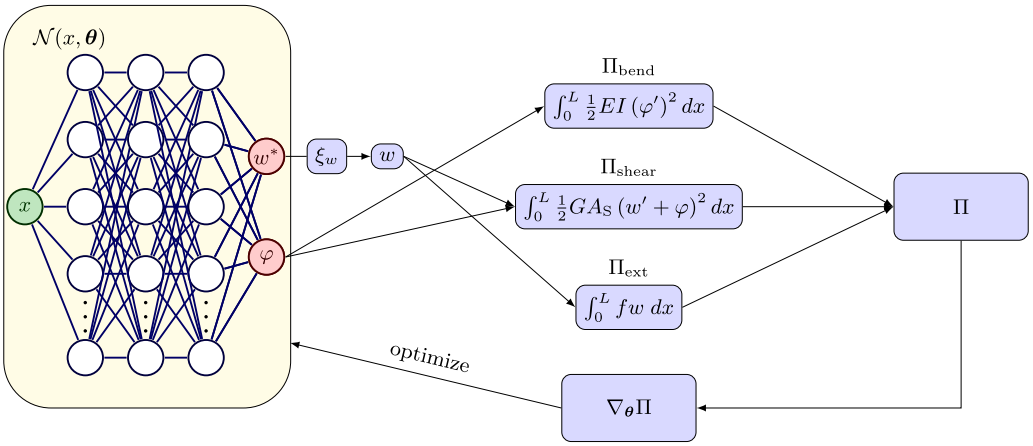


Fig. 1. Schematic of the baseline PINN for the Timoshenko beam derived from the (w, φ)-parametrization. The PINN outputs w^* and φ. The distance function ξ_w is applied to w^* before the components of the potential are evaluated. During training, the gradients $\nabla_{\boldsymbol{\theta}} \Pi$ are calculated and passed to the optimizer.

3.3.3. (w, w_s)-Parametrization

The idea for the (w, w_s)-parametrization [47] is based on a split of the displacement w into a bending part w_b and a shear part w_s

$$w = w_b + w_s, \tag{19}$$

which has its origins in foundational works on beam theory, such as those by [48,49]. Any two of these three displacements can be chosen as primary variables. In this work, we choose w and w_s, due to the application of boundary conditions on w and the constraint on w_s using distance functions, as discussed in Section 2. The total rotation φ and the shear strain γ can then be computed as

$$\varphi = -w_b' = -(w - w_s)' = -w' + w_s' \tag{20}$$

$$\gamma = w_s'. \tag{21}$$

The total potential in terms of the primary variables is thus given as

$$\Pi = \int_0^L \frac{1}{2} EI\,(-w'' + w_s'')^2\,dx + \int_0^L \frac{1}{2} GA_S\,(w_s')^2\,dx - \Pi_{\text{ext}}. \tag{22}$$

The fully balanced kinematic equations in both energy terms avoid shear locking and corresponding shear stress oscillations [47]. When using the (w, w_s)-parametrization, an additional constraint for w_s must be imposed. This can be done by setting $w_s = 0$ at one boundary, which removes the possibility of zero energy modes [39].

4. Shear locking in PINNs

In this section, we analyze shear locking in PINNs using a simply supported Timoshenko beam under constant load as a representative example. The underlying mechanisms, however, are equally relevant to Reissner–Mindlin plates. Our focus is on the gradient structure and training dynamics of the loss function; the full problem setup and numerical results are presented in Section 6.1.

As a baseline PINN, we consider the (w, φ)-parametrization with a single neural network approximating both w and φ, as shown in Fig. 1. The network predicts the unconstrained displacement w^* and the rotation φ. Dirichlet boundary conditions for the displacement are imposed by multiplying w^* by the distance function ξ_w. For this baseline PINN, Dirichlet boundary conditions on the total rotation φ could be imposed analogously. This is no longer straightforward for the hierarchic parametrizations, in which φ is not a primary variable. Consequently, to ensure a fair comparison, we add a penalty term to the loss function for all PINNs when imposing clamped boundary conditions.

For the baseline PINN, shear locking becomes increasingly pronounced as the slenderness increases. Fig. 2 shows the evolution of the relative L_2-error of the displacement w during training. For a beam with slenderness $s = 10$, the error begins to decrease after roughly 100 iterations. For a beam with slenderness $s = 10^3$, by contrast, only a minor reduction in the relative L_2-error is observed within 50,000 iterations.

Fig. 3 shows the prediction for the slender beam. The PINN response is overly stiff, underestimating both the displacement w and the rotation φ. Moreover, the shear force Q is overestimated and exhibits oscillations. These are characteristic symptoms of locking, as discussed previously.

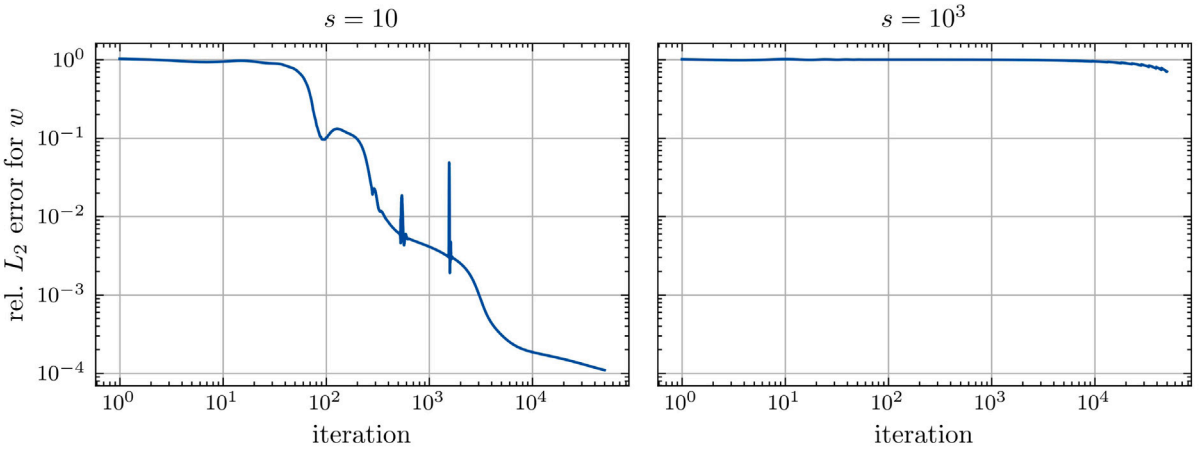


Fig. 2. Relative L_2-error for the displacement w for a thick beam (left) and a slender beam (right) over 50,000 training iterations.

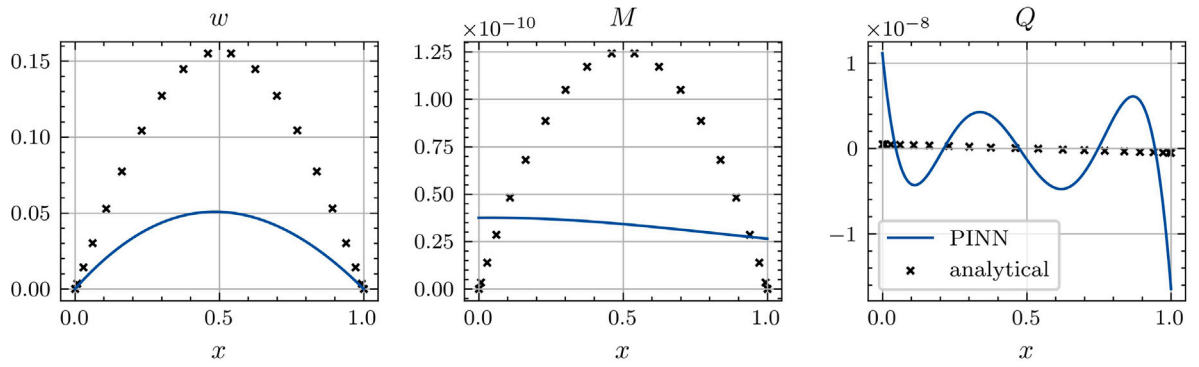


Fig. 3. Prediction of the baseline PINN for slenderness $s = 10^3$, compared with the analytical solution.

The observed locking can be traced to the training dynamics induced by the energy functional and the chosen parametrization. We analyze these dynamics using the gradient-flow equation of the loss function. For the (w, φ)-parametrization, the gradient flow equation is given as

$$\frac{\partial \boldsymbol{\theta}}{\partial t} = -\nabla_\theta \Pi = -\nabla_\theta \left(\int_0^L \frac{1}{2} EI \left(\varphi'\right)^2 \, dx + \int_0^L \frac{1}{2} GA_S \left(w' + \varphi\right)^2 \, dx - \int_0^L f w \, dx \right) \tag{23}$$

$$= -\left(\int_0^L EI \varphi' \nabla_\theta \varphi' dx + \int_0^L GA_S (w' + \varphi)(\nabla_\theta w' + \nabla_\theta \varphi) dx - \int_0^L f \nabla_\theta w \, dx \right), \tag{24}$$

under the assumption that the training depends continuously on the training time t. The gradient-flow equation is not intended to be an exact model of the Adam/AMSGrad iterations used in the following numerical experiments. However, the dominant gradient contributions identified below are properties of the energy functional itself. The gradient norm fractions and cosine similarities reported in the following are evaluated along the actual Adam/AMSGrad training trajectories. Thus, the gradient-flow equations provide an interpretation of the loss landscape and its slenderness-dependent scaling, while the numerical diagnostics verify that the same mechanism is relevant for the optimizer used in this work.

As the slenderness increases, the shear stiffness GA_S becomes much larger than the bending stiffness EI, resulting in the training process being increasingly dominated by the shear energy term.

To illustrate this effect, we introduce a modified potential $\hat{\Pi}$ obtained from Π by removing the bending energy, so that only the shear energy and the external work remain. We then train two identically initialized PINNs and measure the alignment of the gradients $\nabla_\theta \Pi$ and $\nabla_\theta \hat{\Pi}$ by evaluating the cosine similarity

$$\cos\left(\nabla_\theta \Pi, \nabla_\theta \hat{\Pi}\right) = \frac{\nabla_\theta \Pi \cdot \nabla_\theta \hat{\Pi}}{\|\nabla_\theta \Pi\|_2 \|\nabla_\theta \hat{\Pi}\|_2}. \tag{25}$$

A value of 1 indicates that $\nabla_\theta \Pi$ and $\nabla_\theta \hat{\Pi}$ point in the same direction, a value of 0 indicates orthogonality, and a value of -1 indicates opposite directions. The corresponding results are shown in Fig. 4 for a thick and a slender beam. For the thick beam, the

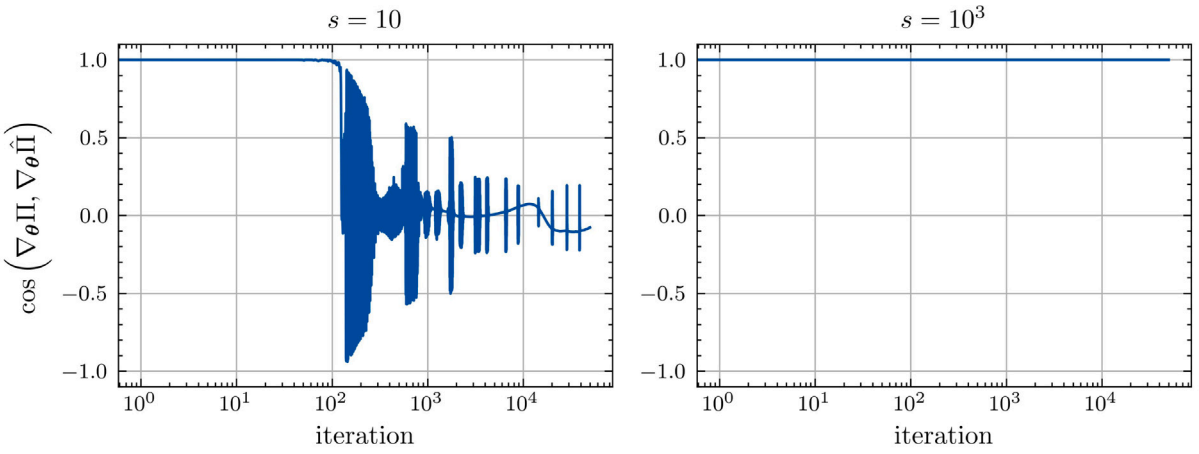


Fig. 4. Cosine similarity between the gradients $\nabla_\theta \Pi$ and $\nabla_\theta \hat{\Pi}$ during training for a thick and a slender beam.

cosine similarity starts at 1 and approaches orthogonality after roughly 100 iterations. For the slender beam, the cosine similarity stays at a value of 1 throughout the full 50,000 iterations. This shows that, in the slender case, the training is entirely governed by the shear energy term.

Further insight can be obtained by comparing the gradient norms of the different contributions to the potential, namely $\|\nabla_\theta \Pi_{\text{bend}}\|_2$, $\|\nabla_\theta \Pi_{\text{shear}}\|_2$, and $\|\nabla_\theta(-\Pi_{\text{ext}})\|_2$. Rather than considering the raw norms, we evaluate the fraction of the total gradient magnitude associated with each term:

$$\alpha_{(\text{bend/shear/ext})} = \frac{\|\nabla_\theta \Pi_{(\text{bend/shear/ext})}\|_2}{\|\nabla_\theta \Pi_{\text{bend}}\|_2 + \|\nabla_\theta \Pi_{\text{shear}}\|_2 + \|\nabla_\theta(-\Pi_{\text{ext}})\|_2}. \tag{26}$$

The results are shown in Fig. 5. For the thick beam, the bending contribution increases from 0% to approximately 25% after roughly 100 iterations, coinciding with the decrease of the displacement error in Fig. 2 and with the transition of the cosine similarity in Fig. 4 toward 0. The contributions α_{shear} and α_{ext} are initially at 65% and 35% and stabilize around 40% and 35% after 100 iterations, indicating an overall balance between the three terms. For the slender beam, by contrast, α_{bend} remains close to 0% for most of the training and increases only near the end to approximately 10%, where minor improvements in the relative L_2-error of w can be observed in Fig. 2. The shear contribution is close to 100% and the contribution of the external work is close to 0% at the beginning of the training. After roughly 100 iterations, α_{shear} starts decreasing while α_{ext} increases until both reach comparable magnitudes of $\alpha_{\text{ext}} \approx 55\%$ and $\alpha_{\text{shear}} \approx 45\%$ after 200 iterations. At this stage, their cosine similarity is close to -1, as shown in Fig. 6, indicating that the corresponding gradients act in opposite directions. This is less pronounced for the thicker beam, the cosine similarity of the shear energy and the external work stabilizes around -0.8 after roughly 100 iterations. Consequently, for the slender beam, the gradient of the external work, which should drive the deformation of the beam, is nearly canceled by the gradient of the shear energy. This results in a local minimum in which the optimizer remains trapped if the bending energy gradient is too small. The resulting solution is overly stiff and absorbs the external work primarily through the shear energy, which is the main mechanism of the locking behavior observed in this setting. This local minimum is reached easily because the shear energy and the external work are strongly coupled in the sense that both depend on the displacement w and its derivative w', whereas the bending energy depends only on the curvature φ', as seen in Eq. (24), and is therefore only indirectly coupled to the external work.

Based on this analysis, we identify two main causes of shear locking in PINNs based on the standard (w, φ)-parametrization:

1. Training is dominated by the shear energy. This becomes more pronounced as $GA_S \gg EI$ for increasing slenderness. Since the bending energy remains too small for most of the training, it provides only a weak driving force toward the bending-dominated response, which leads to slow convergence.
2. The external work is only weakly coupled to the bending energy, in the sense that the bending energy depends only on the rotation, whereas the external work depends on the displacement. In contrast, the shear energy and the external work are strongly coupled, which makes it easy for the optimizer to converge to a local minimum in which the gradient of the external work is canceled by the gradient of the shear energy.

These issues arise from the stiff PDE, its parametrization, and the resulting optimization landscape. Standard loss-balancing and gradient-scaling techniques commonly used in the PINN literature [50–52] are not applicable here, because they effectively assign weights to different loss terms of the form

$$\mathcal{L}_\lambda = \lambda_{\text{PDE}} \mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}} \mathcal{L}_{\text{BC}} + \lambda_{\text{data}} \mathcal{L}_{\text{data}}, \tag{27}$$

where each term represents a separate loss function, which is zero at the solution, and thus can be scaled independently by a factor λ.

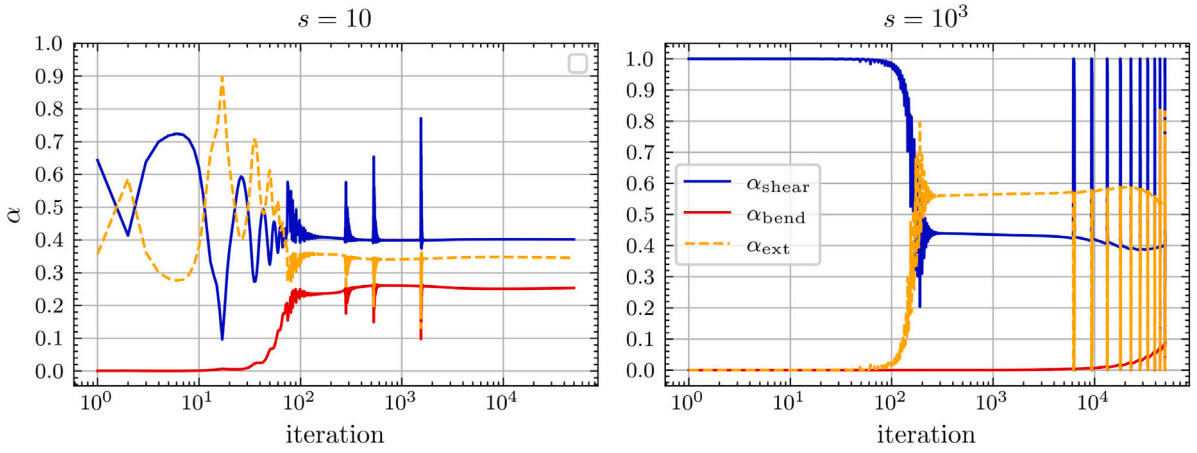


Fig. 5. Fraction of the total gradient magnitude contributed by each term during training.

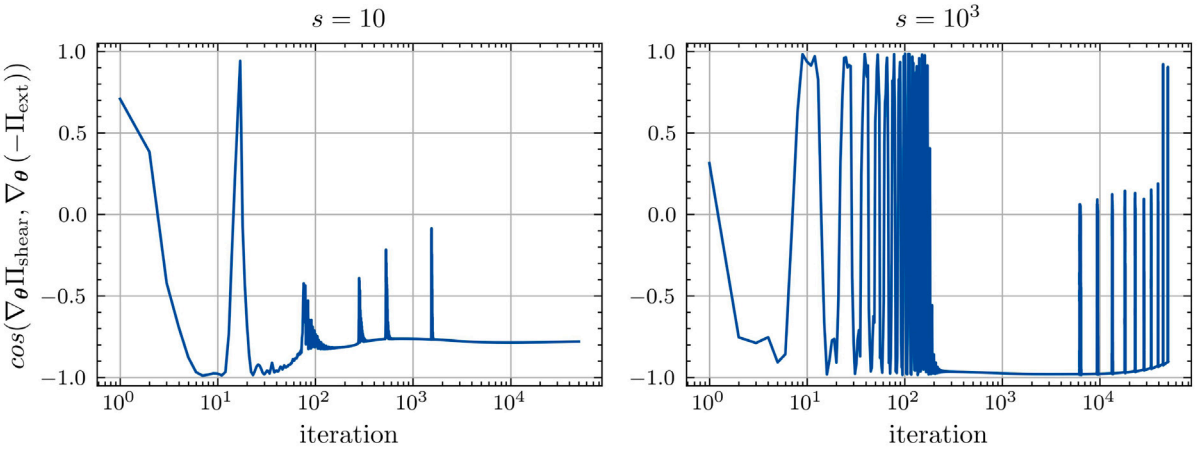


Fig. 6. Cosine similarity between $\nabla_\theta \Pi_{\text{shear}}$ and $\nabla_\theta \Pi_{\text{ext}}$.

In this work, the loss function is the total potential energy, computed from the bending energy, the shear energy and the external work. The total potential energy is minimized as a whole, but the individual components are not zero at the solution. Therefore, the individual components cannot be scaled independently, as this would change the underlying physical problem. This is a fundamental difference to the standard loss-balancing and gradient-scaling techniques, which are not applicable in this setting. This is why the strategy pursued below changes the parametrization of the unknown fields and the scaling of the shear variable, while leaving the physical energy functional unchanged.

5. Shear-locking-free PINNs based on hierarchic parametrizations

Section 4 identified the optimization landscape as the main source of the observed shear-locking behavior for the energy-based PINNs. This differs from the standard explanation in FEM and IGA, where locking is attributed to imbalanced function spaces, as discussed in Sections 3.3.1–3.3.3. Classical discretization methods determine their best approximation by solving a system of equations and rely on comparatively low-dimensional approximation spaces. In the present PINN setting, the neural networks provide more flexible function approximators [53,54], and our numerical results indicate that the optimization landscape induced by the energy functional and the chosen parametrization is the dominant limitation.

For PINNs, hierarchic parametrizations offer two main advantages. First, the direct parametrization of a shear variable enables targeted control of the gradients associated with the shear term. Second, the bending energy becomes more strongly coupled to the external work in the sense that both depend on the displacement, which reduces the tendency of the shear term to cancel the deformation-driving gradients of the external work. For the (w, γ)-parametrization, the gradient-flow equation is

$$-\nabla_\theta \Pi = -\left[\int_0^L EI\left(\gamma' - w''\right)\left(\nabla_\theta \gamma' - \nabla_\theta w''\right)\, dx + \int_0^L GA_S \gamma \nabla_\theta \gamma \, dx - \int_0^L f \nabla_\theta w dx\right], \tag{28}$$

whereas for the (w, w_s)-parametrization it is

$$-\nabla_\theta \Pi = -\left[\int_0^L EI\left(-w'' + w_\mathrm{s}''\right)\left(-\nabla_\theta w'' + \nabla_\theta w_\mathrm{s}''\right)\, dx + \int_0^L GA_\mathrm{S} w_\mathrm{s}' \nabla_\theta w_\mathrm{s}'\, dx - \int_0^L f \nabla_\theta w dx\right]. \tag{29}$$

In both cases, the bending energy and the external work depend on the displacement w, while the shear energy is isolated in the shear variable γ or w_s, which does not appear in the external work. This yields a more favorable optimization landscape and directly addresses the second cause of locking identified in Section 4. To fully exploit this advantage, it is important to recall that the network parameters were previously collected in a single variable $\boldsymbol{\theta}$ for notational simplicity. In a fully connected neural network, only the weights and biases of the last layer act exclusively on one output neuron, whereas all preceding layers are shared between the primary variables. Consequently, most parameter updates remain influenced by the large shear gradients. To avoid this coupling, we employ two separate neural networks, each predicting a single primary variable. This yields two separate parameter sets, $\boldsymbol{\theta}_w$ and $\boldsymbol{\theta}_\gamma$ or $\boldsymbol{\theta}_{w_\mathrm{s}}$, according to

$$\mathcal{N}_w(x, \boldsymbol{\theta}_w) = w, \quad \mathcal{N}_\gamma(x, \boldsymbol{\theta}_\gamma) = \gamma, \tag{30}$$

$$\mathcal{N}_w(x, \boldsymbol{\theta}_w) = w, \quad \mathcal{N}_{w_\mathrm{s}}(x, \boldsymbol{\theta}_{w_\mathrm{s}}) = w_\mathrm{s}. \tag{31}$$

With this modification, the displacement w can converge independently of the slenderness of the beam or plate. The shear variable γ or w_s, however, still suffers from the large magnitude of the shear energy gradients. To address this, we scale the output of the network predicting the shear variable. More precisely, the network predicts γ^* or w_s^*, from which the physical shear strain γ or shear displacement w_s is recovered by multiplication with a factor β:

$$\mathcal{N}_\gamma(x, \boldsymbol{\theta}_\gamma) = \gamma^*, \quad \gamma = \gamma^* \beta, \tag{32}$$

$$\mathcal{N}_{w_\mathrm{s}}(x, \boldsymbol{\theta}_{w_\mathrm{s}}) = w_\mathrm{s}^*, \quad w_\mathrm{s} = w_\mathrm{s}^* \beta. \tag{33}$$

To determine the scaling factor β, we consider the gradient-flow equation for the shear parameters. For the (w, γ)-parametrization, this gives

$$-\nabla_{\theta_\gamma} \Pi = -\left[\int_0^L EI\left(-w'' + \beta\gamma^{*'}\right)\beta \nabla_{\theta_\gamma}\gamma^{*'} + GA_\mathrm{S}\beta^2\gamma^* \nabla_{\theta_\gamma}\gamma^*\, dx\right] \tag{34}$$

$$= -\left[\int_0^L -EIw''\beta\nabla_{\theta_\gamma}\gamma^{*'} + EI\beta^2\gamma^{*'}\nabla_{\theta_\gamma}\gamma^{*'} + GA_\mathrm{S}\beta^2\gamma^*\nabla_{\theta_\gamma}\gamma^*\, dx\right], \tag{35}$$

and for the (w, w_s)-parametrization

$$-\nabla_{\theta_{w_\mathrm{s}}} \Pi = -\left[\int_0^L -EIw''\beta\nabla_{\theta_{w_\mathrm{s}}} w_\mathrm{s}^{*''} + EI\beta^2 w_\mathrm{s}^{*''}\nabla_{\theta_{w_\mathrm{s}}} w_\mathrm{s}^{*''} + GA_\mathrm{S}\beta^2 w_\mathrm{s}^{*'}\nabla_{\theta_{w_\mathrm{s}}} w_\mathrm{s}^{*'}\, dx\right]. \tag{36}$$

For both parametrizations, three terms remain. Since the bending stiffness EI is proportional to s^{-3} and the shear stiffness GA_S to s^{-1}, the influence of the slenderness and the scaling factor β on the magnitude of the gradients can be estimated based on the simplified expression

$$As^{-3}\beta + Bs^{-3}\beta^2 + Cs^{-1}\beta^2. \tag{37}$$

A good balance of these terms is obtained for the scaling $\beta = s^{-2}$, which yields the slenderness-dependent magnitudes

$$As^{-5} + Bs^{-7} + Cs^{-5}. \tag{38}$$

This choice places the shear energy term and the more important bending-related term on comparable scales across all slendernesses. Choosing $\beta = s^{-1}$ or $\beta = s^{-3}$ would instead bias the training toward the shear or bending term, respectively. The argument should therefore be understood as an order-of-magnitude estimate for the shear-variable scaling: $\beta = s^{-2}$ balances the dominant slenderness-dependent gradient contributions, but is not claimed to be unique or optimal for the full optimization problem. An additional benefit of this scaling is that the initial values of the shear strain γ approach zero for slender beams, which is consistent with the Bernoulli hypothesis, without requiring an explicit modification of the network initialization.

The resulting PINNs for the Timoshenko beam are shown in Fig. 7 for the (w, γ)-parametrization and in Fig. 8 for the (w, w_s)-parametrization. Together, the hierarchic parametrization, the use of separate neural networks, and the scaling of the shear variable resolve both issues identified in Section 4. The bending energy is directly coupled to the external work, the trainable parameters of the primary variables are separated, and the shear-related gradients are balanced through the scaling of the shear variable. Hierarchic formulations can therefore be used effectively to mitigate shear locking in PINNs, but only when adapted to the optimization-driven locking mechanism described previously.

Fig. 9 shows the fractions of the gradient norms for each network in the modified (w, γ)-formulation, evaluated for the same problem considered in Section 4. For both the thick and the slender beam, the gradients of the network $\mathcal{N}_w$ receive comparably sized contributions from the bending energy and the external work within 100 iterations. The network predicting the shear variable receives contributions from both the bending and shear terms, while the initial shear share increases only mildly from $\alpha_\mathrm{shear} \approx 0.8$ to $\alpha_\mathrm{shear} \approx 0.9$ as the slenderness increases from 10 to 10^3. As a result, the training becomes essentially independent of the slenderness, as demonstrated in the numerical experiments below.

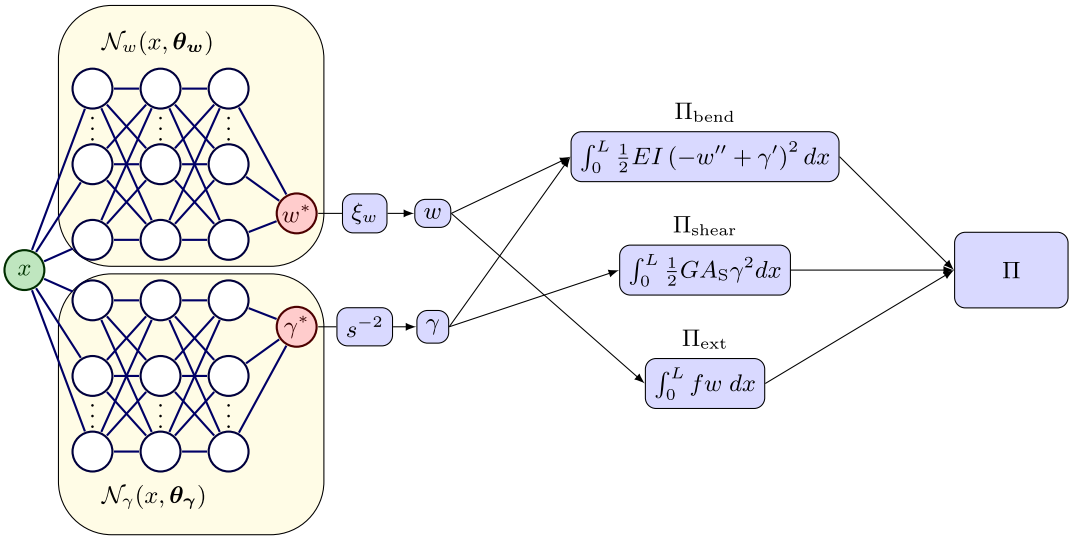


Fig. 7. Schematic of the PINN for the Timoshenko beam based on the (w, γ)-parametrization with two neural networks and scaling of the shear variable. The PINN outputs w^* and γ^*. The distance function is applied to w^* before the components of the potential are evaluated.

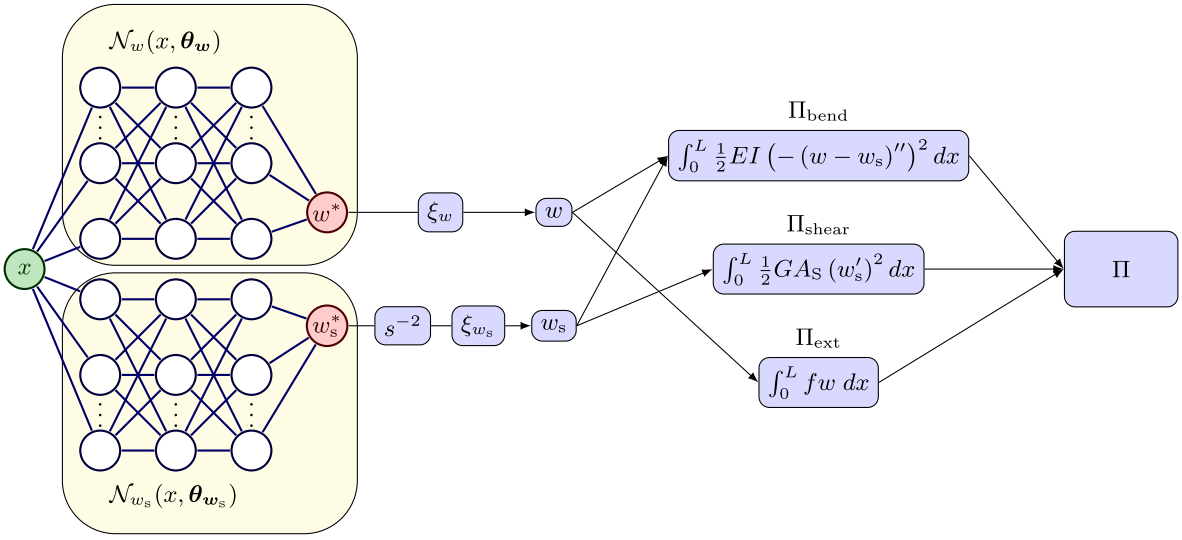


Fig. 8. Schematic of the PINN for the Timoshenko beam based on the (w, w_s)-parametrization with two neural networks and scaling of the shear variable. The PINN outputs w^* and w_s^*. A distance function is applied to w^* and w_s^* before the components of the potential are evaluated.

6. Numerical tests

This section presents numerical tests for the proposed PINNs for Timoshenko beams and Reissner–Mindlin plates. To assess the occurrence of shear locking, we compare training speed and accuracy for different parametrizations over a range of slenderness values from 10 to 10^4. Accuracy is measured by the relative L_2-error of the relevant variables with respect to the analytical solution,

$$\text{relative } L_2\text{-error} = \frac{\|y_{\text{PINN}} - y_{\text{exact}}\|_2}{\|y_{\text{exact}}\|_2}. \tag{39}$$

We consider a linear elastic material with Young's modulus $E = 1$ and Poisson's ratio $\nu = 0.3$.

Models based on two separate neural networks use two hidden layers with 32 neurons per layer and the GELU activation function [55]. For PINNs based on a single neural network, the width is increased to 46 neurons per layer in order to obtain a comparable number of trainable parameters. These hyperparameters are chosen based on the preliminary study reported in Appendix A. In agreement with previous observations in the literature [12,13], the exact choice of hyperparameters has little influence on

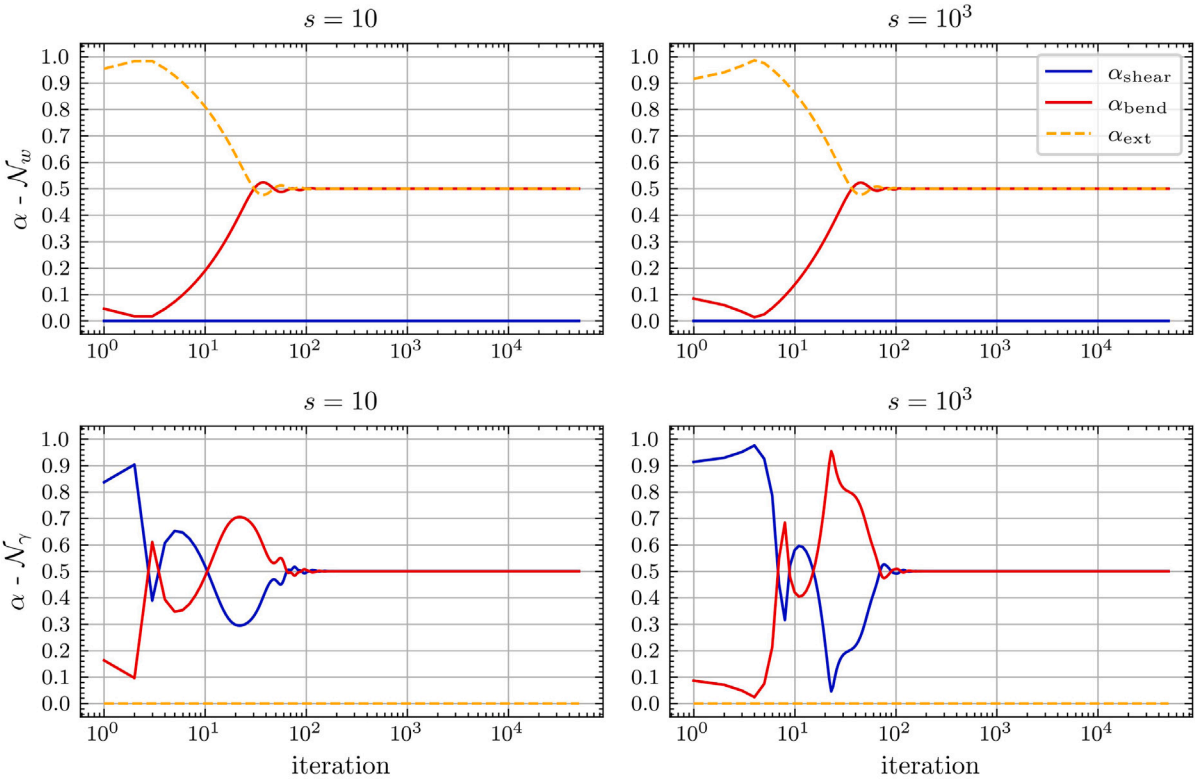


Fig. 9. Magnitude fractions of the gradient norms for the (w, γ)-parametrization with two neural networks and scaling of the shear variable for slenderness $s = 10$ (left) and $s = 10^3$ (right). The gradients are evaluated for the neural network predicting the displacement w (top) and for the neural network predicting the shear strain γ (bottom).

the qualitative behavior, provided that the network is sufficiently large. In particular, within the parameter range considered, the locking behavior persists independently of the chosen hyperparameters, indicating that it is not solely caused by insufficient network capacity.

The PINNs are implemented in PyTorch [56] using double-precision floating-point arithmetic. For all examples, we use the Adam optimizer [57] with the AMSGrad variant [58] and set $\epsilon = 0$. This avoids damping of the parameter updates for the very slender structures, where the energies become very small and the default value $\epsilon = 10^{-8}$ would be too large. Training is performed for 50,000 iterations.

All examples are posed on input domains that already lie in a suitable range for the networks, so no input normalization is required. The last layer uses the identity activation and therefore acts as a linear regression layer. We use the default initialization for linear layers in PyTorch, i.e., Kaiming uniform initialization for the weights [59] and zero initialization for the biases.

The required integrals are evaluated by Gauss quadrature. The exact role of numerical integration in energy-based PINNs remains an open question and is currently under active investigation [60], since quadrature errors may induce forward divergence, a distinct failure mode of energy-based PINNs identified in [61]. Appendix B studies the influence of the number of integration points for the beam example in Section 6.1. Unless stated otherwise, we use 128 integration points for the beam examples and 128^2 quadrature points for the plate examples.

To account for the random nature of the initialization of the neural network parameters, each training run is repeated ten times with random initializations. We report average results as well as a shaded area indicating the minimum and maximum values observed during the ten runs.

6.1. Simply supported beam under constant load

We consider a simply supported beam of length $L = 1$, subjected to a constant load $f = Et^3$. The analytical solutions for the displacement $w(x)$ and the rotation $\varphi(x)$ are given by

$$w(x) = \frac{f}{24EI}\left(L^3 x - 2Lx^3 + x^4\right) + \frac{f}{2GA_S}\left(Lx - x^2\right), \tag{40}$$

$$\varphi(x) = -\frac{f}{24EI}\left(L^3 - 6Lx^2 + 4x^3\right). \tag{41}$$

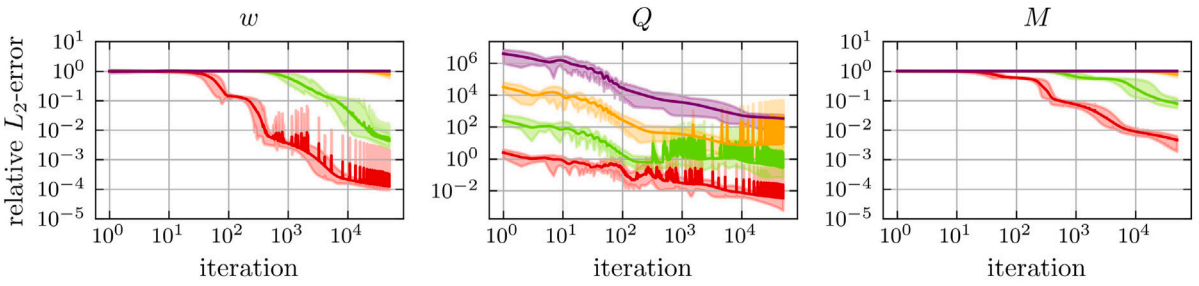


Fig. 10. Relative L_2-errors for the (w, φ)-parametrization of the simply supported Timoshenko beam under a constant load using a single neural network without further modifications. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

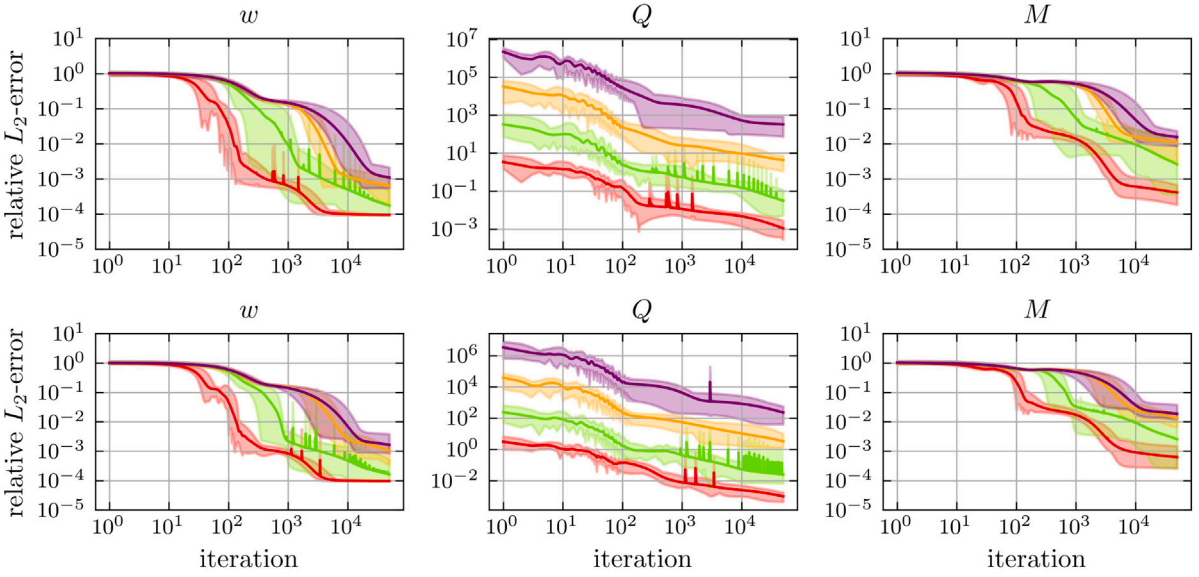


Fig. 11. Relative L_2-errors for the (w, γ)- (top) and the (w, w_s)- (bottom) parametrization of the simply supported Timoshenko beam under a constant load using a single neural network without scaling the shear deformation variable. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

For the implementation of the Dirichlet boundary conditions and constraints, we use the following functions:

$$\xi_w(w^*, x) = w^* x(L - x), \tag{42}$$

$$\xi_{w_s}(w_s^*, x) = w_s^* x. \tag{43}$$

The slenderness is varied by changing the thickness t of the beam, while keeping the length fixed at $L = 1$.

Fig. 10 shows the relative L_2-errors for the displacement w, the bending moment M and the shear force Q for the (w, φ)-parametrization using a single neural network with no further adjustments (as depicted in Fig. 1). For the beam with a slenderness of 10, 50,000 iterations are required to reduce the error for the displacement w to approximately 10^{-4} and for the bending moment M and shear force Q to approximately 10^{-2}. For the beam with a slenderness of 100, we can observe a high initial error for the shear force Q of approximately 10^2, which is reduced to an error of approximately 10^{-1} after 50,000 iterations. The displacement w and the bending moment M show a plateau for the first 1000 iterations, after which the error starts slowly decreasing to approximately 10^{-2} for the displacement and 10^{-1} for the bending moment, indicating that the shear energy is dominant in the loss function. For more slender beams, this behavior becomes even more pronounced. The initial errors for Q increase and the errors for w and M remain almost unchanged within the 50,000 iterations.

In Fig. 11, the results for the (w, γ)- and (w, w_s)-parametrization using a single neural network and no scaling of the shear deformation variable are shown. The initial errors for the shear force Q remain high for slender beams, which leads to a low convergence rate during training. We can observe an improvement in the convergence rate of the displacement w and the bending moment M for slender beams, but the performance still degrades with increasing slenderness. The improvement in convergence rate compared to the (w, φ)-parametrization is due to the direct parametrization of the shear strain γ (or the shear displacement w_s). For these formulations, the displacement w and the bending moment M are less influenced by the shear energy term, because

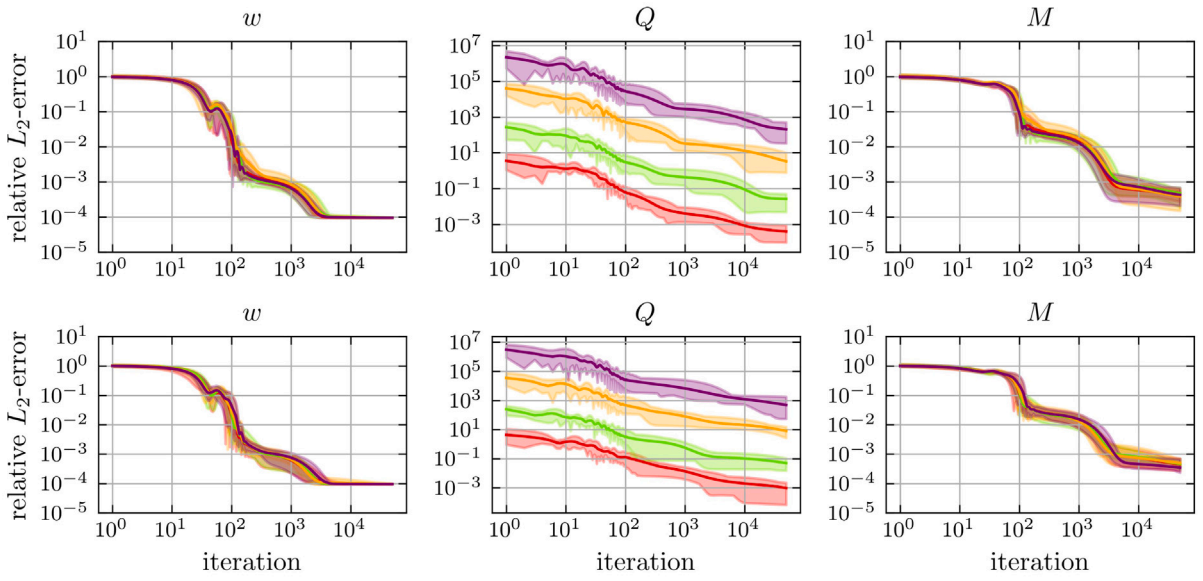


Fig. 12. Relative L_2-errors for the (w,γ)- (top) and (w,w_s)- (bottom) parametrization of the simply supported Timoshenko beam under a constant load using two neural networks without scaling of the shear deformation variable. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

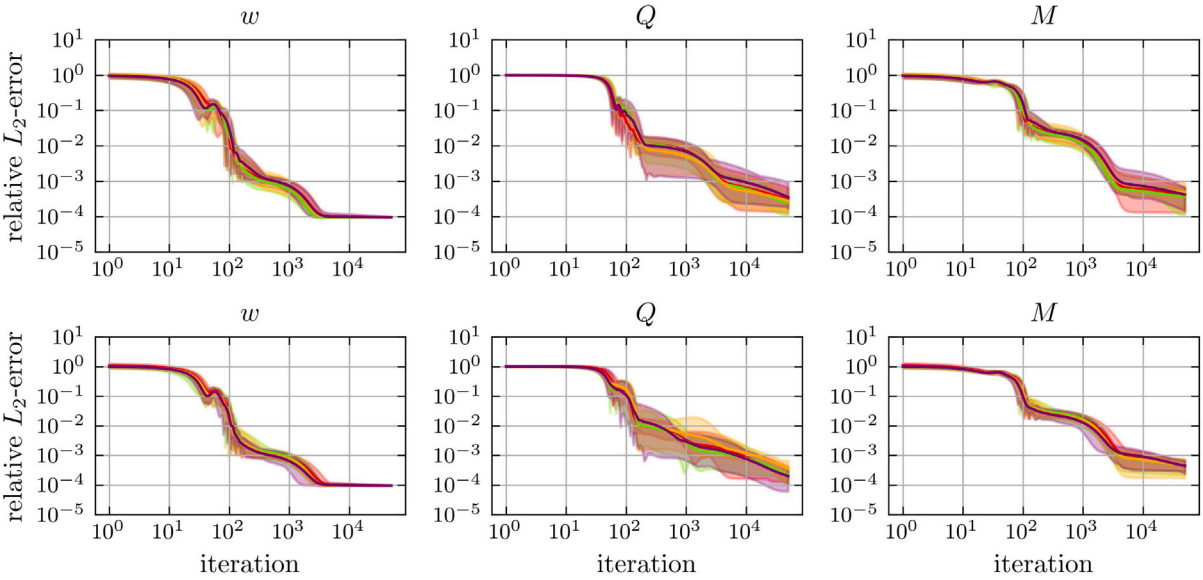


Fig. 13. Relative L_2-errors for the (w,γ)- (top) and the (w,w_s)- (bottom) parametrization of the simply supported Timoshenko beam under a constant load using two neural networks with scaling of the shear deformation variable. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

the shear energy term is calculated solely based on the shear deformation variable γ or w_s. The gradients of the shear energy affect only the shared trainable parameters of the neural network, i.e. the weights and biases before the last layer. We can see that naively implementing hierarchic formulations for PINNs does not fully remove the locking behavior; further adjustments are required.

To fully remove locking, separate neural networks for each primary variable can be used, which results in a full separation of the trainable parameters for each variable. The results for these PINNs are shown in Fig. 12. We can see that for both formulations, the displacement w and the bending moment M converge independently of the slenderness of the beam. However, the initial errors for the shear force Q remain high for slender beams, which leads to high errors in the shear force Q even after 50,000 iterations.

In Fig. 13, the global errors for the (w,γ)- and the (w,w_s)-parametrization using two separate neural networks and scaling of the shear deformation variable are shown, as depicted in Figs. 7 and 8 for the (w,γ)- and (w,w_s)-parametrization, respectively. We can see that all relative L_2-errors converge independently of the slenderness of the beam. The scaling of the shear deformation

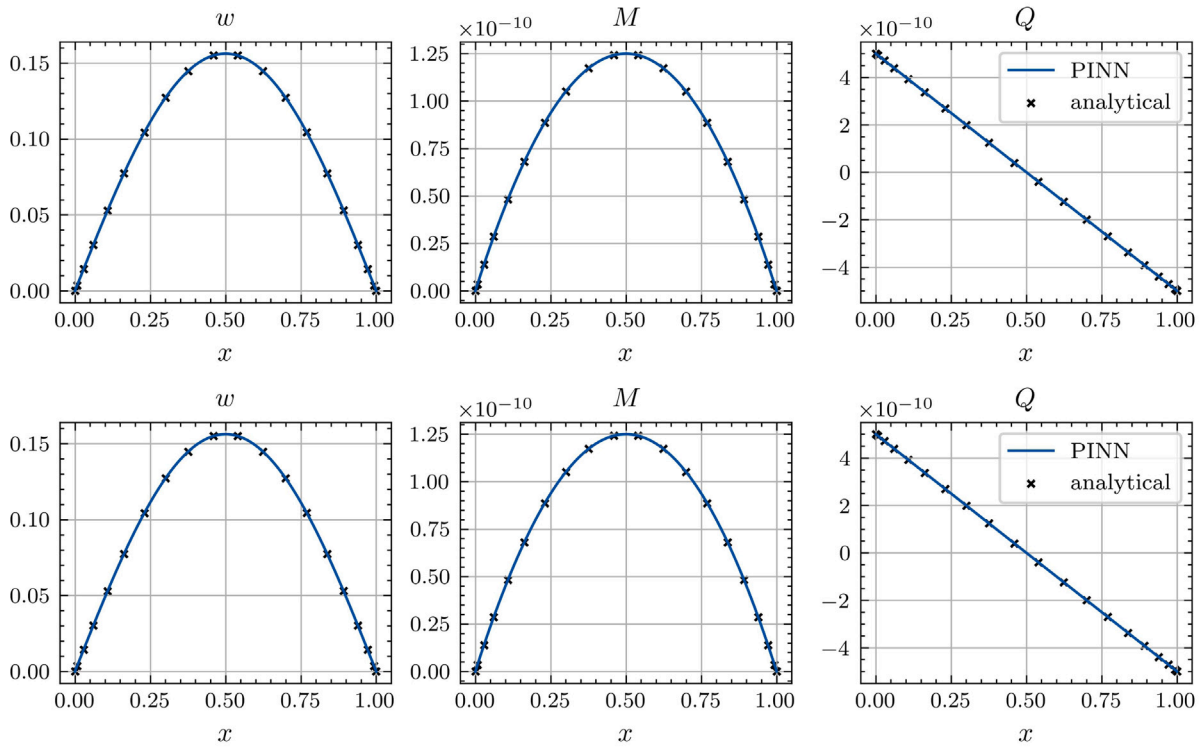


Fig. 14. Predictions of the modified (w, γ)-PINN (top) and the modified (w, w_s)-PINN (bottom) compared to the analytical solution for a slenderness of $s = 10^3$.

variable leads to a suitable initial ratio of the shear and the bending energy terms. The relative errors for the displacement w reduce to approximately 10^{-4} within approximately 5000 iterations for all slendernesses considered. In comparison to the baseline PINN based on the (w, φ)-parametrization shown in Fig. 10, this is a substantial improvement even for the lowest slenderness of 10, where the baseline PINN only reached a relative error of approximately 10^{-4} for the displacement w after 50,000 iterations. For more slender beams, the improvement is even more pronounced, as the baseline PINN did not converge at all within the 50,000 iterations for a slenderness of 10^3 and 10^4. For the shear force Q and the bending moment M, similar gains in convergence rate can be observed, with both errors converging to approximately 10^{-3} within less than 10,000 iterations for all slendernesses considered.

In Fig. 14, the predictions of the modified (w, γ)- and (w, w_s)-parametrization are compared to the analytical solution for a slenderness of $s = 10^3$. Comparing this to the result of the baseline PINN in Fig. 3, it is clear that the locking has been resolved. The displacement and the stress resultants are captured accurately. Oscillations in the shear force are removed for both parametrizations. The (w, γ)- and the (w, w_s)-parametrization show similar training behavior. The balance of the derivatives of primary variables in the bending energy, which is advantageous in the IGA context, as discussed in Section 3.3.3, appears to be irrelevant for the PINNs. The (w, w_s)-parametrization is more expensive to compute, because it requires second-order derivatives of w and of w_s, while the (w, γ)-parametrization only requires second-order derivatives of w and first-order derivatives of γ. Consequently, the (w, γ)-parametrization should be preferred when implementing Timoshenko beam PINNs.

6.2. Clamped–clamped beam under constant load

As a next test, we consider a beam of unit length $[0, 1]$ clamped at both ends and subjected to the load $f = Et^3$. The analytical solutions for the displacement and rotation are given by

$$w(x) = \frac{f}{24EI}(x^4 - 2x^3 + x^2) + \frac{f}{2GA_S}(-x^2 + x), \tag{44}$$

$$\varphi(x) = -\frac{f}{12EI}(2x^3 - 3x^2 + x). \tag{45}$$

To enforce the boundary conditions for the displacement and the constraint on the shear deformation, we use the same distance functions as in the previous example. To enforce the rotational boundary conditions, we add a weighted penalty term for the rotations at the two clamped ends:

$$\mathcal{L} = \Pi + \frac{\lambda}{2}\left(\varphi(0)^2 + \varphi(1)^2\right). \tag{46}$$

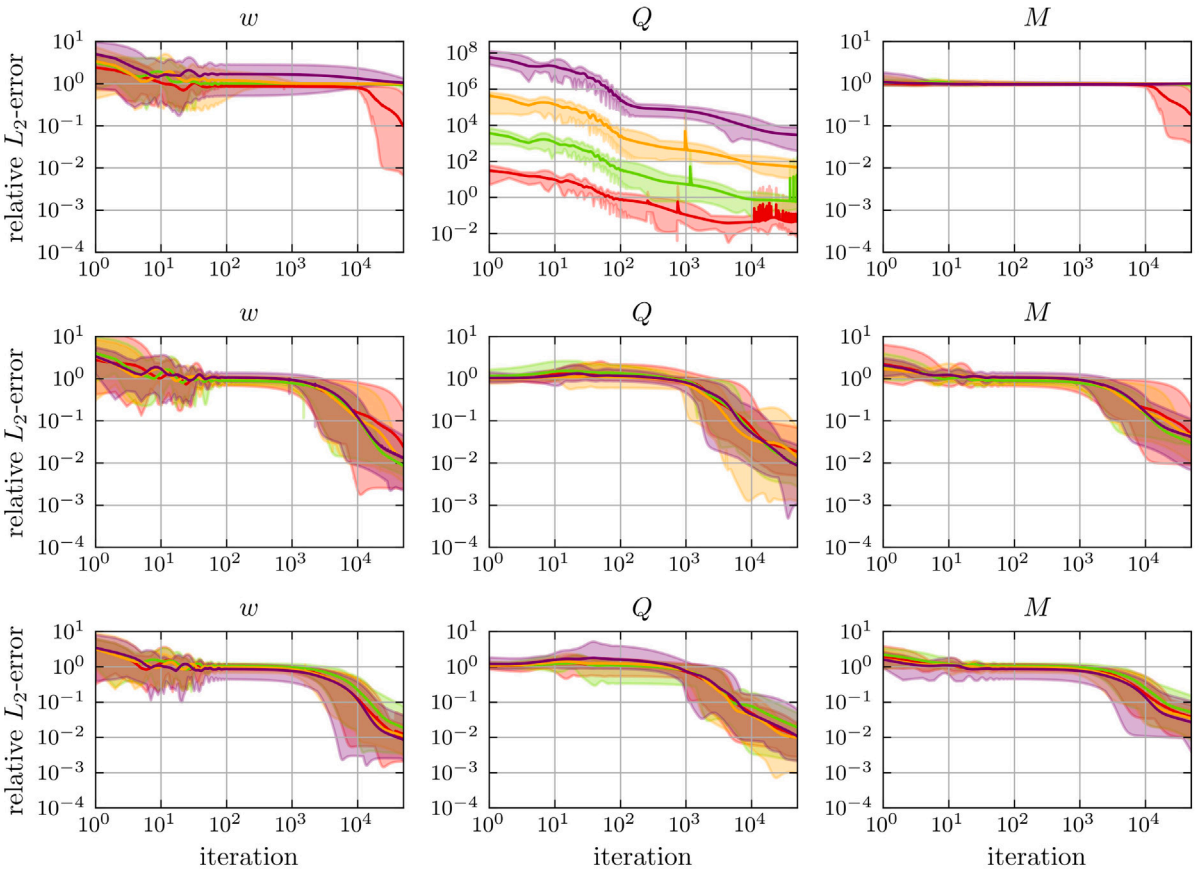


Fig. 15. Relative L_2-errors for the baseline PINN (top), the (w, γ)-parametrization (middle) and the (w, w_s)-parametrization (bottom) for the clamped–clamped beam under constant load. The models based on the hierarchic parametrizations use separate neural networks and the scaling of the shear variable. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

We found a weighting factor of $\lambda = 500t^3$ to provide the best balance of accuracy and training efficiency. The factor is scaled proportionally to the thickness t^3 to account for the varying magnitude of the total potential.

The results for the baseline PINN as well as the locking-free PINNs based on the hierarchic parametrizations with separate networks and scaling of the shear variable are shown in Fig. 15. For the baseline PINN, convergence of the displacement and bending moment toward the reference solution is observed only for the thickest beam, and only after approximately 10,000 iterations. The final accuracies are on the order of 10^{-1} for the displacement and the bending moment. The shear force exhibits increasing errors with increasing slenderness.

For the locking-free PINNs, all slenderness ratios exhibit similar convergence rates. For both parametrizations, the errors in the displacement and shear force decrease to approximately 10^{-2}, while the bending moment error reduces to about $5 \cdot 10^{-2}$ within 50,000 iterations.

Compared to the previous example, all PINNs exhibit higher errors. This is expected due to the weak imposition of the clamped boundary conditions. The training process shows a plateau during the first 1000 iterations, possibly due to the increased stiffness of the loss function caused by the additional penalty term. These effects highlight the need to develop methods for strongly enforcing clamped boundary conditions for the hierarchic parametrizations in PINNs, in order to improve both accuracy and convergence speed in these scenarios.

6.3. Circular Reissner–Mindlin plate

We consider a circular plate with radius $R = 1$, which is simply supported at its outer edge and subjected to a constant load $f = Et^3$. The analytical solutions for the displacement $w(r)$ and the radial rotation $\varphi_r(r)$ as functions of the radial coordinate r are given in [62]:

$$w(r) = \frac{fR^4}{64D}\left(\frac{5+\nu}{1+\nu} + \left(\frac{r}{R}\right)^4 - 2\frac{3+\nu}{1+\nu}\left(\frac{r}{R}\right)^2\right) + \frac{fR^2}{4kGt}\left(1 - \left(\frac{r}{R}\right)^2\right),$$

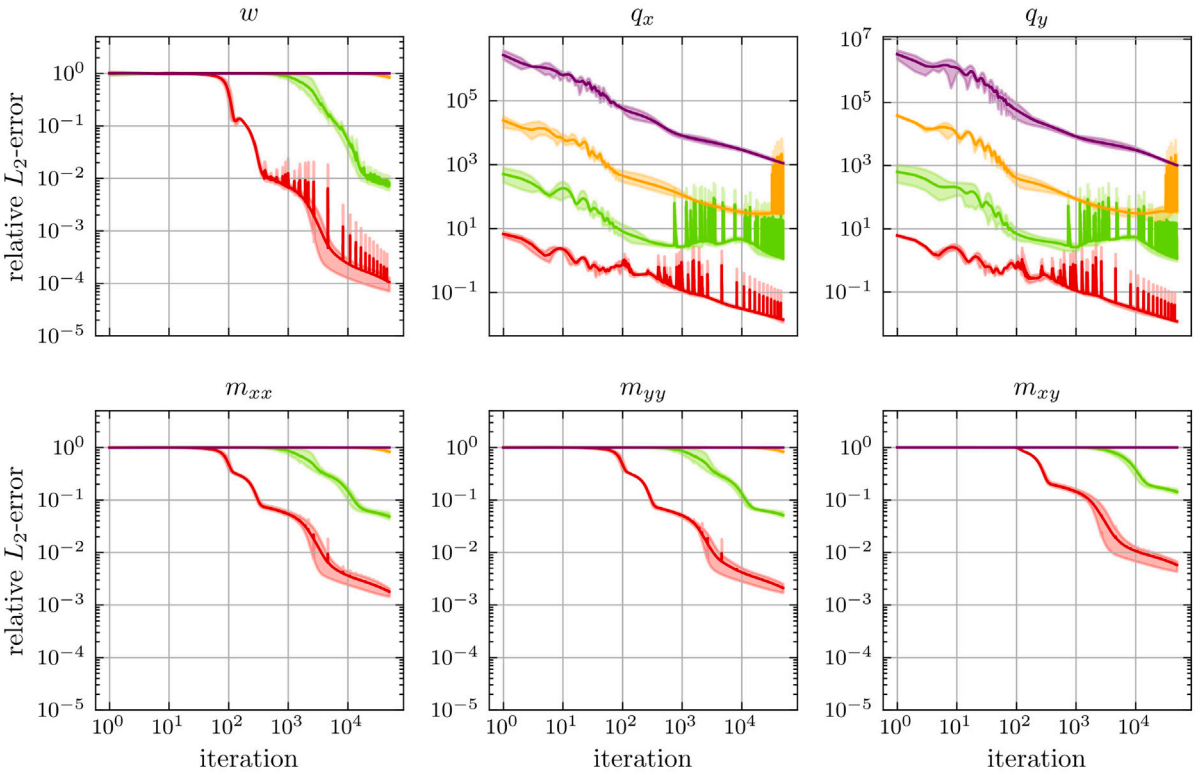


Fig. 16. Relative L_2-errors for the (w, φ)-parametrization of the circular plate under a constant load using a single neural network without further modifications as a baseline PINN. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

$$\varphi_r(r) = \frac{f R^3}{16D} \frac{r}{R} \left(1 - \left(\frac{r}{R}\right)^2\right).$$

While the problem is axisymmetric, it should be noted that the PINN is not informed about this fact and treats the problem as a two-dimensional problem in Cartesian coordinates.

For the implementation of the Dirichlet boundary conditions and constraints, we use the following functions:

$$\xi_w(w^*, x, y) = w^* \left(\frac{x^2 + y^2}{R^2} - 1\right), \tag{47}$$

$$\xi_{w_{s_1}}(w^*_{s_1}, x) = w^*_{s_1} (x - R), \tag{48}$$

$$\xi_{w_{s_2}}(w^*_{s_2}, y) = w^*_{s_2} (y - R). \tag{49}$$

In Fig. 16, the relative L_2-errors for the displacement w, the moment components m_{xx}, m_{yy} and m_{xy} and the shear force components q_x and q_y for the (w, φ)-parametrization using a single neural network to predict all primary variables are shown. Similar to the Timoshenko beam results shown in Fig. 10, the performance is poor. When the slenderness increases, an increasingly pronounced plateau at the beginning of the training for the displacement w and the components of the moment tensor m_{xx}, m_{yy} and m_{xy} can be observed, as well as high initial errors for the shear forces. For PINNs with a slenderness greater than 100, no change of the errors of the displacement and the moments can be observed within the 50,000 iterations.

We skip the intermediate PINN models, as they show similar behavior to the Timoshenko beam results discussed in Section 6.1. Instead, we directly present the results for the PINNs based on two separate neural networks and scaling of the shear variables. For these PINNs, one network predicts the displacement w, while the other network predicts both shear strains γ_x, γ_y or both components of the shear displacement w_{s_1}, w_{s_2}. A further separation into three networks, one for each primary variable, does not lead to relevant improvements, because a full separation of the trainable parameters for both shear components is not required to mitigate locking. Furthermore, using three neural networks instead of two increases the computational cost by approximately 10% in our setup. In Fig. 17 and Fig. 18, the results for these proposed PINNs are shown. Similar to the Timoshenko beam results shown in Fig. 11, we can see that all relative L_2-errors converge independently of the slenderness of the plate. The displacement w reaches a relative error of approximately 10^{-4} within 10,000 iterations for all slendernesses considered. For the moment components m_{xx} and m_{yy}, similar gains in convergence rate can be observed, with all errors converging to approximately 10^{-3} within less than 10,000 iterations for all slendernesses considered. The shear force components q_x, q_y and the twisting moment m_{xy} reach an error

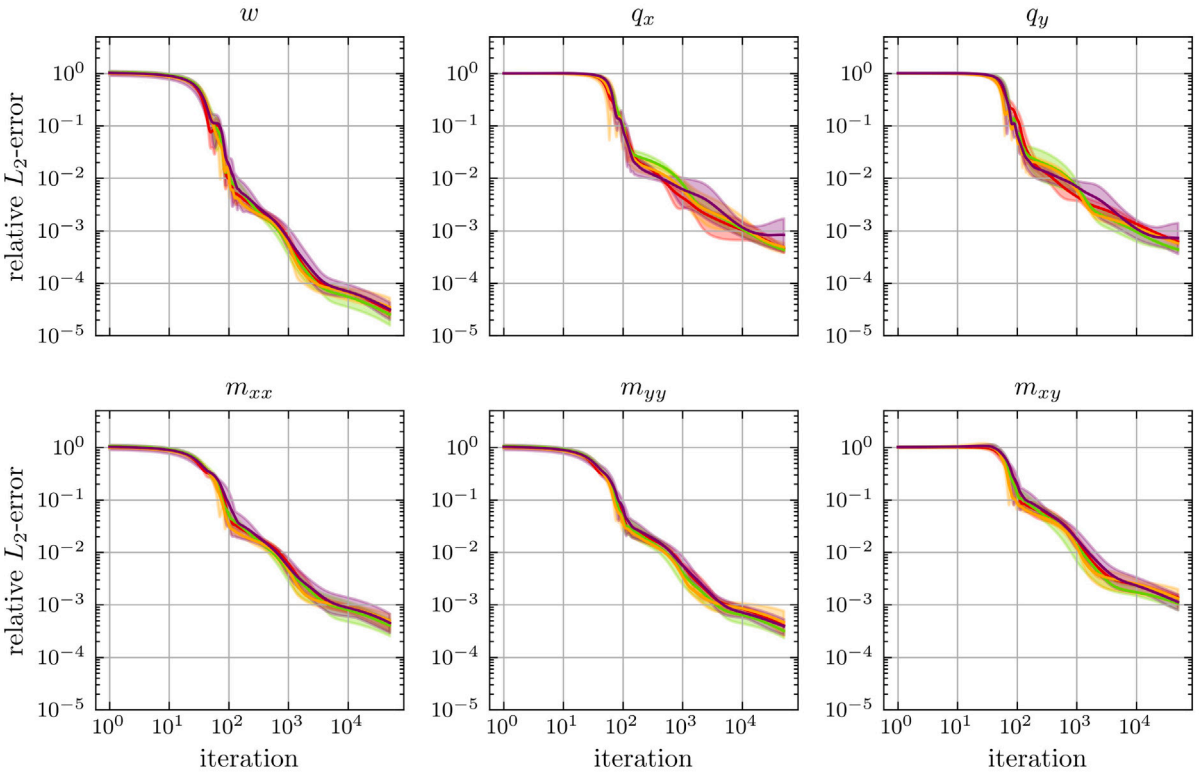


Fig. 17. Relative L_2-errors for the (w,γ)-parametrization of the circular plate under a constant load using two neural networks with scaling of the shear deformation variables. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

of approximately 10^{-3} within the 50,000 iterations for all slendernesses considered. These results demonstrate that the proposed methods for mitigating shear locking in PINNs based on the hierarchic formulations are effective not only for Timoshenko beams, but also for Reissner–Mindlin plates.

Comparing the results of the (w,γ)-parametrization and the (w,w_s)-parametrization, again no significant differences can be observed. As such, the choice between the two formulations can be made based on considerations of computational efficiency and ease of implementation. The difference in the order of derivatives required to compute the potential energy makes the (w,γ)-parametrization more efficient, especially in the two-dimensional domain of plates, which results in approximately 50% reduced time per optimizer iteration in our setup. Furthermore, the (w,γ)-parametrization does not require the imposition of additional boundary conditions for the shear deformations w_{s_1} and w_{s_2} and is consequently easier to implement.

6.4. Clamped square plate subjected to distributed load

As a further example, we consider a clamped unit square plate $[0,1]^2$ subjected to the distributed load

$$f = \frac{Et^3}{12(1-\nu^2)}\Big[\,12y(y-1)(5x^2-5x+1)\left(2y^2(y-1)^2+x(x-1)(5y^2-5y+1)\right) + 12x(x-1)(5y^2-5y+1)\left(2x^2(x-1)^2+y(y-1)(5x^2-5x+1)\right)\Big]. \tag{50}$$

The analytical solution for the displacement is given in [63], while the corresponding bending moments and shear forces are derived in [36]. They are given by

$$w = \frac{1}{3}x^3(x-1)^3y^3(y-1)^3 - \frac{2t^2}{5(1-\nu)}\left[y^3(y-1)^3x(x-1)(5x^2-5x+1)\right. \tag{51}$$

$$\left. + x^3(x-1)^3y(y-1)(5y^2-5y+1)\right], \tag{52}$$

$$q_x = -\frac{Et^3}{6(1-\nu^2)}\left[y^3(y-1)^3(20x^3-30x^2+12x-1)+3y(y-1)(5y^2-5y+1)x^2(x-1)^2(2x-1)\right], \tag{53}$$

$$q_y = -\frac{Et^3}{6(1-\nu^2)}\left[x^3(x-1)^3(20y^3-30y^2+12y-1)+3x(x-1)(5x^2-5x+1)y^2(y-1)^2(2y-1)\right], \tag{54}$$

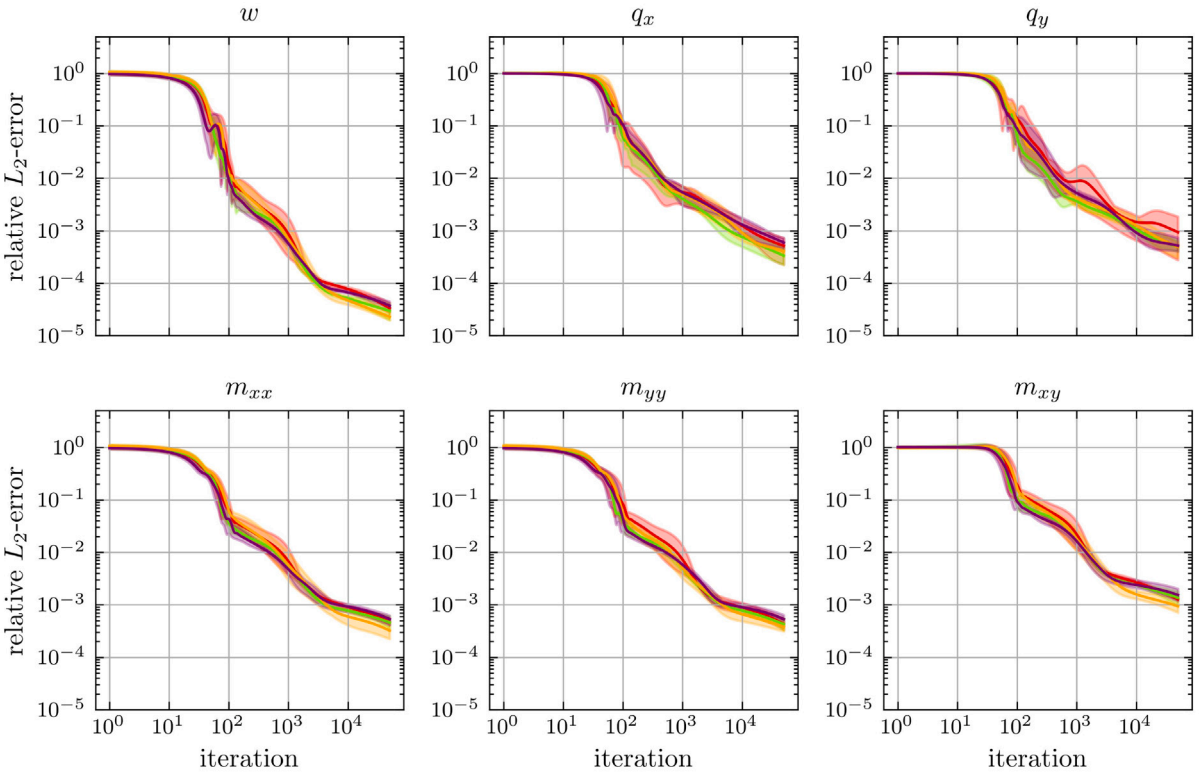


Fig. 18. Relative L_2-errors for the (w, w_s)-parametrization of the circular plate under a constant load using two neural networks with scaling of the shear deformation variables. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

$$m_{xx} = \frac{Et^3}{6(1-\nu^2)}\left[y^3(y-1)^3(x-x^2)(5x^2-5x+1) + \nu x^3(x-1)^3(y-y^2)(5y^2-5y+1)\right], \tag{55}$$

$$m_{yy} = \frac{Et^3}{6(1-\nu^2)}\left[\nu y^3(y-1)^3(x-x^2)(5x^2-5x+1) + x^3(x-1)^3(y-y^2)(5y^2-5y+1)\right], \tag{56}$$

$$m_{xy} = -\frac{Et^3}{4(1+\nu)} y^2(y-1)^2(2y-1)x^2(x-1)^2(2x-1). \tag{57}$$

To impose the displacement boundary conditions, we use the distance functions

$$\xi_w(w^*, x, y) = w^* x(x-1)y(y-1), \tag{58}$$

$$\xi_{w_{s_1}}(w^*_{s_1}, x) = w^*_{s_1} x, \tag{59}$$

$$\xi_{w_{s_2}}(w^*_{s_2}, y) = w^*_{s_2} y. \tag{60}$$

To impose the boundary conditions on the rotations, we uniformly sample $n = 400$ points in total on all edges of the plate and compute the mean square error of the rotations as an additional term in the loss function:

$$\mathcal{L} = \Pi + \frac{\lambda}{n}\sum_{i=1}^{n}\left(\varphi_x(x_i, y_i)^2 + \varphi_y(x_i, y_i)^2\right). \tag{61}$$

We set the weighting factor $\lambda = t^3$ for this example to provide the best balance of accuracy and training efficiency.

For this example, we observed divergence caused by quadrature errors. To resolve this issue, we increased the number of quadrature points from 128^2 to 256^2.

Fig. 19 shows the results for the baseline PINN based on the (w, φ)-parametrization. The PINN again exhibits severe locking. For a slenderness of 10, the relative L_2-error decreases to approximately $5 \cdot 10^{-4}$ for the displacement and to approximately 10^{-2} for both the bending moments and the shear forces. As the slenderness increases, however, the convergence rate deteriorates substantially. For slenderness values greater than 100, the errors in the displacement and bending moments remain essentially unchanged over the full training.

Fig. 20 shows the results using the (w, γ)-parametrization with two neural networks and the scaling of the shear variables. In this case, the errors begin to decrease independently of the slenderness of the plate. A slight reduction in convergence rate of the

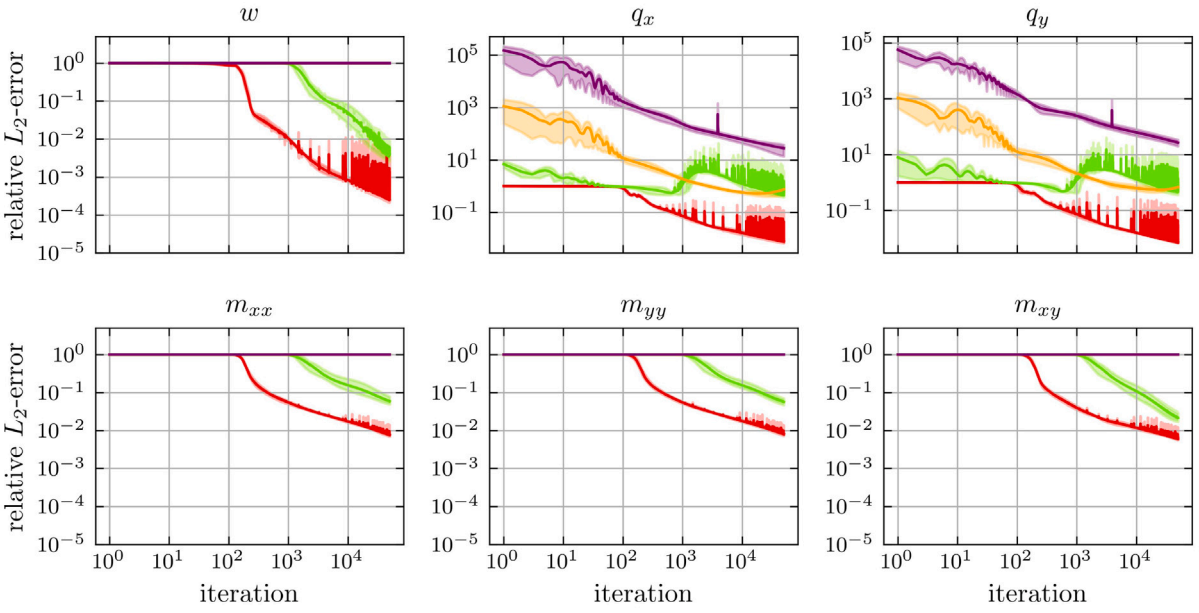


Fig. 19. Relative L_2-errors for the (w, φ)-parametrization of the square plate subjected to a distributed load using a single neural network without further modifications. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

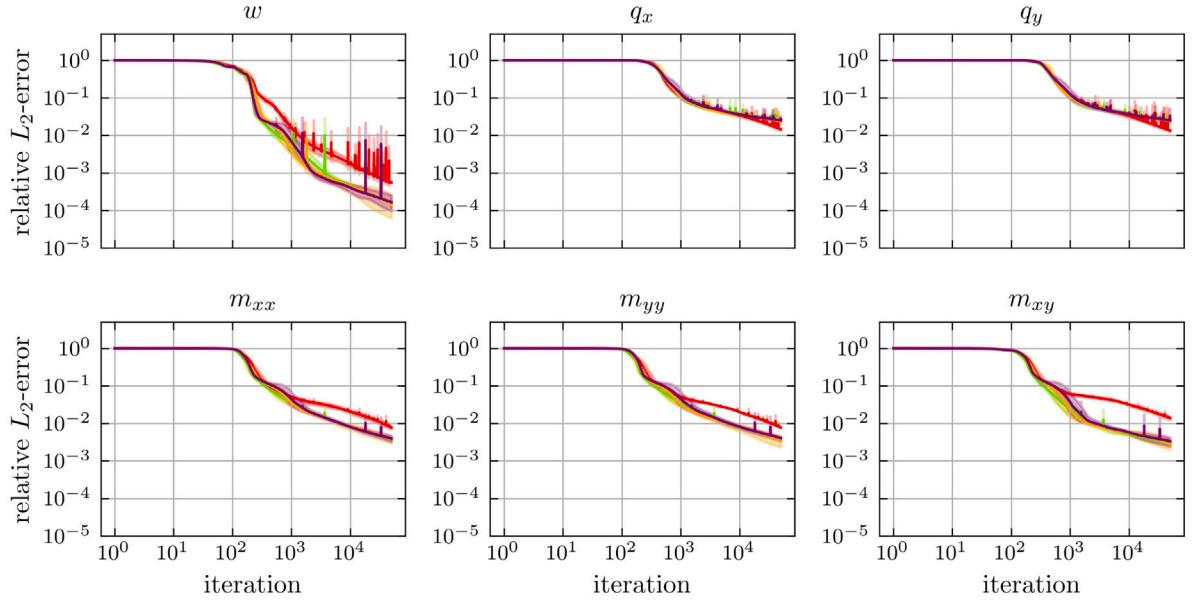


Fig. 20. Relative L_2-errors for the (w, γ)-parametrization of the square plate subjected to a distributed load using two neural networks with scaling of the shear deformation variables. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

displacement and the bending moments is observed for the thickest plate. Overall, the errors are larger than in the circular plate example in Section 6.3, possibly due to the weak imposition of boundary conditions or the higher complexity of the load function and solution fields.

The corresponding results for the (w, w_s)-parametrization are shown in Fig. 21. The overall behavior is again very similar to that of the (w, γ)-parametrization. However, for the shear forces, more slender plates exhibit a reduced convergence rate and instabilities toward the end of training. These effects may be alleviated by increasing the number of integration points or the weighting factor of the boundary term. In our experiments, the (w, w_s)-parametrization is more susceptible to these failure modes, underscoring the practical advantages of the (w, γ)-parametrization.

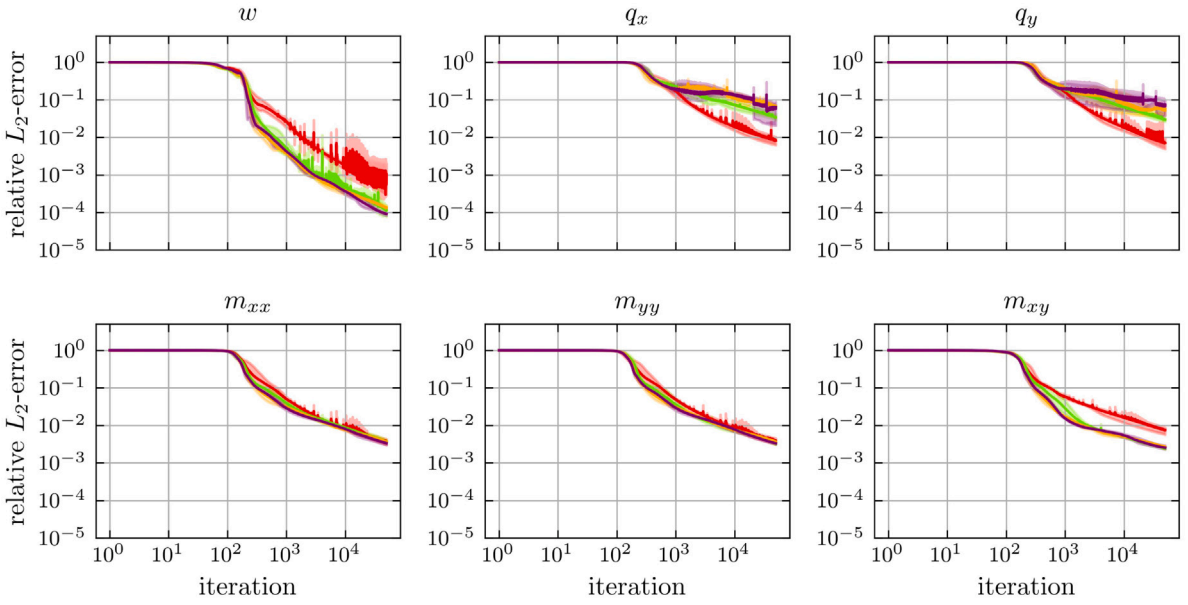


Fig. 21. Relative L_2-errors for the (w, w_s)-parametrization of the square plate subjected to a distributed load using two neural networks with scaling of the shear deformation variables. Legend: $s = 10$, $s = 10^2$, $s = 10^3$, $s = 10^4$.

6.5. Parametric PINN for the circular plate

To illustrate transfer to a related PDE-loss-based learning framework, we consider a parametric PINN based on the P^{2}INN architecture introduced in [24] for the circular plate problem from Section 6.3. In this architecture, the PDE parameters are encoded by a branch network into an intermediate state h_{param}, while a trunk network encodes the spatial coordinates into a state h_{coord}. These embeddings are stacked and passed to a decoder network that outputs the solution values. As the varying PDE parameter, we choose the plate thickness $t \in (10^{-1}, 10^{-4})$.

A schematic of the P^{2}INN based on the (w, φ)-parametrization is shown in Fig. 22. The branch network takes the thickness t as the input and the trunk network takes the spatial coordinates (x, y). Their embeddings are stacked and passed to the decoder network $\mathcal{N}_{(w,\varphi)}$, which predicts all three primary variables. After applying the distance function ξ_w, the resulting fields are used to evaluate the potential for each generated sample. As a locking-free variant, we implement a P^{2}INN based on the (w, γ)-parametrization, shown in Fig. 23. Instead of a single decoder predicting all variables, this architecture uses two decoders, $\mathcal{N}_w$ and $\mathcal{N}_{\gamma_x,\gamma_y}$. We use two hidden layers for each network with 32 neurons per layer for the (w, γ)-P^{2}INN and 38 neurons per layer for the (w, φ)-P^{2}INN to obtain comparable numbers of trainable parameters. The hidden layers use the GELU activation function. In addition, the slenderness-dependent scaling is applied to the shear strains for the (w, γ)-P^{2}INN. For training, 128 problem instances are sampled from the thickness range using logarithmic-uniform sampling in order to represent all thickness scales equally.

Training is performed in full-batch mode, with the batch size equal to the number of sampled problems, for 10,000 iterations. Since the scale of the potential varies strongly with the plate thickness, directly averaging the potentials over all samples would bias the training toward thicker plates. To weight all samples equally, we multiply the potential of each sample by the normalization factor DR^4/f^2. This factor follows from balancing the bending energy density $D(w/R^2)^2$ with the external load density fw, which yields the characteristic deflection scale $w \sim fR^4/D$ and, consequently, the characteristic energy scale f^2R^4/D. For fixed R, the inverse sample-dependent normalization is therefore proportional to D/f^2. The resulting loss function is

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \frac{D}{f_i^2}\Pi_i, \tag{62}$$

where n denotes the number of sampled problems.

Due to the load scaling $f = Et^3$, this is a comparatively mild parametric benchmark, since the displacement varies only weakly with the thickness t. Nevertheless, the example illustrates that the proposed approach can be transferred to more general PINN frameworks that rely on similar PDE-based loss terms.

The trained PINNs are evaluated over the full thickness range, and the relative L_2-errors for the displacement, shear forces, and bending moments are shown in Fig. 24. The parametric PINN based on the (w, γ)-parametrization achieves relative L_2-errors of approximately 10^{-3} for the displacement and the bending moments over the full range of the thickness t. For the shear forces, the errors are slightly larger with a magnitude of 10^{-2}. By contrast, the P^{2}INN based on the (w, φ)-parametrization does not produce meaningful results for any of the quantities considered. The relative L_2-errors of the displacement and bending moments remain close to 1 throughout the thickness range, while the relative error of the shear force increases to approximately 10^4 as the plate becomes

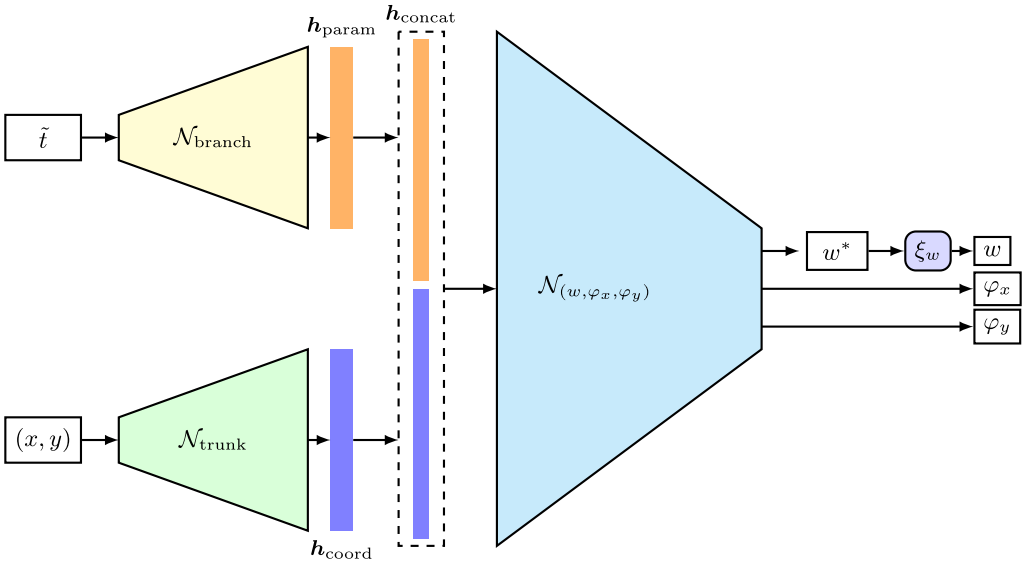


Fig. 22. Schematic of the P^{2}INN based on the (w, φ)-parametrization.

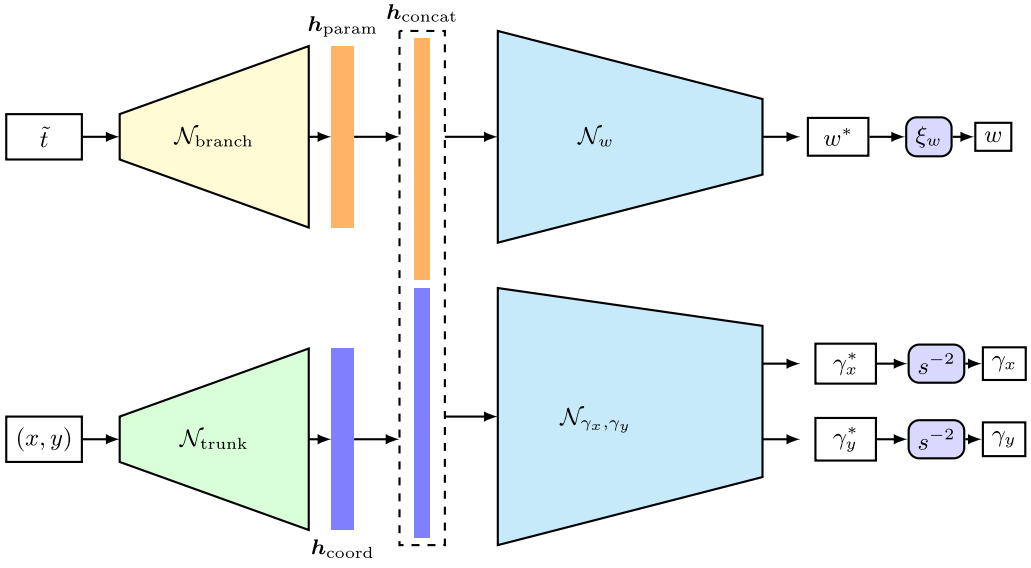


Fig. 23. Schematic of the P^{2}INN based on the (w, γ)-parametrization with two separate decoders and scaling of the shear variables.

thinner. This indicates that in this setup, the shear gradients of the thin plates steer the training due to their large magnitude, which suppresses even the learning of the thicker plates, which could previously be solved in the simple PINN setting.

Predicted fields for a thick plate with $t = 0.1$ and a thin plate with $t = 10^{-4}$ are shown in Figs. 25 and 26, respectively. The P^{2}INN based on the (w, φ)-parametrization underestimates the displacement and bending moments and exhibits oscillatory shear-force predictions for the thin plate, which is consistent with locking. In the present setup, locking degrades the results even for thicker plates, leading to poor overall accuracy across the parameter range. By contrast, the (w, γ)-based P^{2}INN predicts the solution accurately for both the thick and the thin plate.

7. Conclusion

In this paper, we investigated shear locking in energy-based PINNs for shear-deformable beams and plates. The baseline PINNs considered in this work, which are based on the standard parametrization with displacement and rotations as primary variables, exhibit pronounced locking behavior, manifested by slow convergence and poor accuracy for slender structures. Our analysis indicates that this behavior is primarily caused by the optimization landscape induced by the energy functional and the chosen

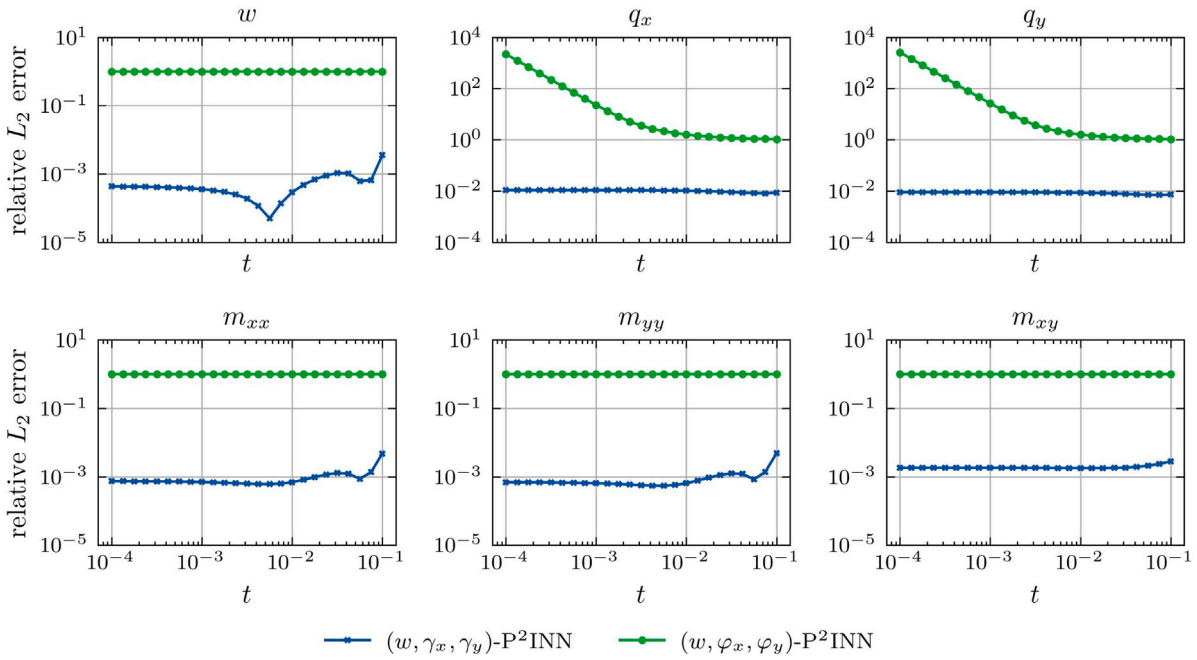


Fig. 24. Relative L_2-errors of the P^{2}INN over the thickness range.

parametrization: as the slenderness increases, the shear term dominates the training dynamics and suppresses the bending-dominated solution.

To mitigate this effect, we considered hierarchic formulations in which the shear contribution is represented by its own primary variable rather than being reconstructed from displacement and rotations. This isolates the shear term and enables a separation of the trainable parameters associated with bending and shear when separate neural networks are used for the primary variables. In addition, we introduce a slenderness-dependent scaling of the shear variable, which balances the relevant terms in the gradient flow and substantially improves the training dynamics. Together, these modifications yield fast and essentially slenderness-independent convergence for all quantities considered in the numerical tests.

In contrast to observations from IGA, where the shear displacement formulation is often superior to the shear strain formulation due to more favorable approximation properties, we do not observe substantial differences between the two hierarchic formulations in the PINN setting. Both exhibit similar convergence behavior and final accuracy. The formulation based on shear displacements is, however, more susceptible to forward divergence and requires more integration points in practice. Since the formulation based on shear strains also requires lower-order derivatives in the potential, it should be preferred for PINNs on grounds of computational efficiency and implementation simplicity.

The present findings may also be relevant beyond the classical PINN framework. The parametric PINN example suggests that the same parametrization strategy can be useful in related PDE-loss-based learning frameworks. Extensions to other architectures, such as physics-informed neural operators, remain a subject for future work.

Future work should address the imposition of rotational boundary conditions without relying on penalty methods to improve convergence speed and accuracy for clamped boundary conditions. It would also be of interest to extend the proposed ideas to strong-form PINNs and to shell models, where membrane locking must be addressed in addition to shear locking.

CRediT authorship contribution statement

Till Weithoff: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Lukas Striefler:** Writing – review & editing, Validation, Methodology, Conceptualization. **Bastian Oesterle:** Writing – review & editing, Supervision, Resources, Conceptualization. **Josef Kiendl:** Writing – review & editing, Supervision, Resources, Conceptualization.

Acknowledgments

We acknowledge financial support by the University of the Bundeswehr Munich.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

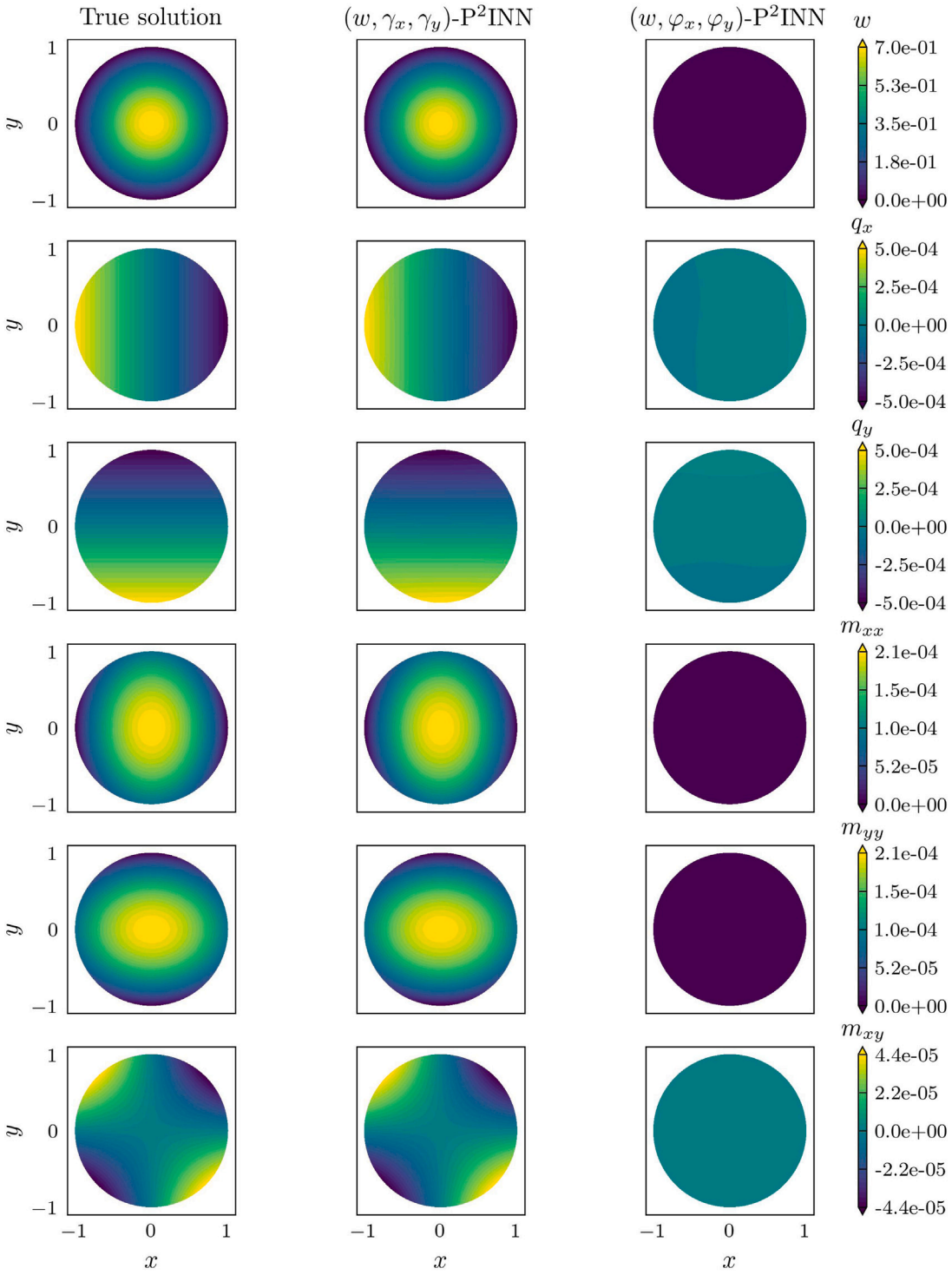


Fig. 25. Solution and predictions of the (w, φ)- and (w, γ)-based P^{2}INNs for a thickness of $t = 0.1$.

Appendix A. Hyperparameter study

To select suitable hyperparameters for the neural networks used in this work, we conduct a parameter study based on the modified (w, γ)-parametrization for the simply supported beam under constant load introduced in Section 6.1. The study varies the

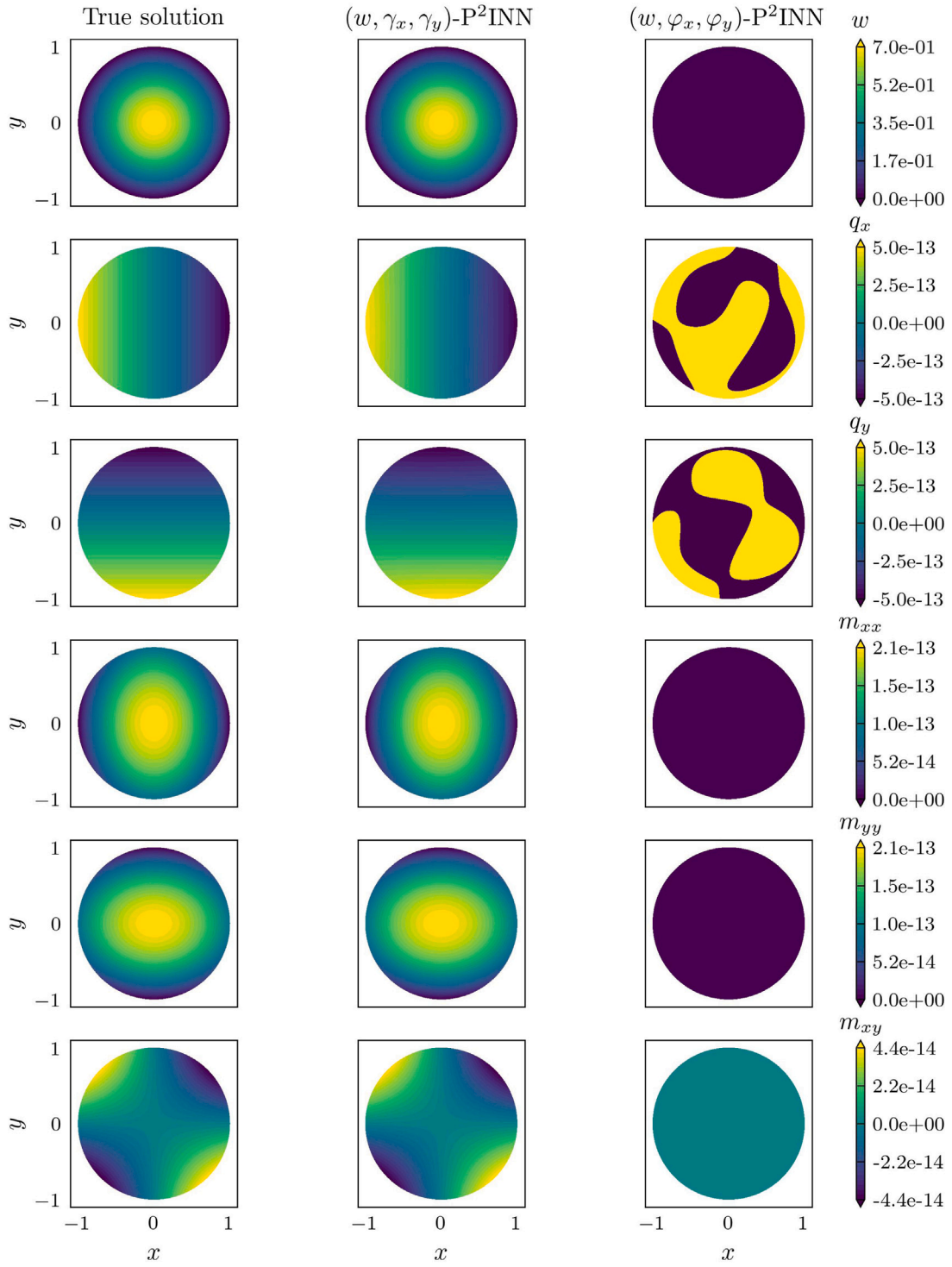


Fig. 26. Solution and predictions of the (w, φ)- and (w, γ)-based P^{2}INNs for a thickness of $t = 10^{-4}$.

activation function, the number of neurons per layer, and the number of hidden layers. All combinations of the settings listed in Table A.1 are considered, resulting in 24 network architectures.

The final relative L_2-errors for each configuration are reported in Table A.2, sorted by the relative L_2-error of the displacement w. All configurations achieve acceptable accuracy. In general, the errors decrease as the depth and width of the networks increase.

Table A.1
Parameters used in the hyperparameter study.

Parameter	Settings
activation	Tanh, GELU
number of neurons	4, 8, 16, 32
number of hidden layers	1, 2, 3

Table A.2
Results of the hyperparameter study.

Number of hidden layers	Neurons per hidden layer	Activation	rel. L_2 w	rel. L_2 Q	rel. L_2 M
2	32	GELU	9.5e−05	3.2e−04	2.1e−04
2	16	GELU	9.7e−05	1.4e−03	1.4e−03
3	16	GELU	9.8e−05	7.7e−04	1.2e−03
3	32	GELU	9.8e−05	6.6e−04	5.2e−04
3	8	GELU	1.0e−04	6.0e−03	1.0e−03
2	32	Tanh	1.5e−04	5.4e−03	3.0e−03
1	32	GELU	1.6e−04	2.1e−03	2.5e−03
3	16	Tanh	1.6e−04	1.1e−02	4.6e−03
3	4	Tanh	2.1e−04	1.3e−02	4.6e−03
3	32	Tanh	2.7e−04	3.8e−03	5.7e−03
2	16	Tanh	3.1e−04	6.4e−03	6.7e−03
1	16	Tanh	5.2e−04	4.5e−03	1.1e−02
1	8	GELU	5.4e−04	3.1e−03	1.0e−02
1	16	GELU	5.8e−04	1.4e−03	1.1e−02
2	8	GELU	6.5e−04	2.8e−03	1.3e−02
1	4	GELU	7.2e−04	1.3e−03	1.4e−02
2	4	GELU	8.1e−04	7.1e−03	1.6e−02
3	4	GELU	8.2e−04	6.1e−03	1.4e−02
3	8	Tanh	9.3e−04	5.0e−03	1.8e−02
2	8	Tanh	1.1e−03	9.8e−03	2.2e−02
1	32	Tanh	1.3e−03	2.9e−03	2.6e−02
1	4	Tanh	1.6e−03	7.2e−03	3.1e−02
2	4	Tanh	2.6e−03	1.7e−02	4.8e−02
1	8	Tanh	2.7e−03	8.9e−03	5.4e−02

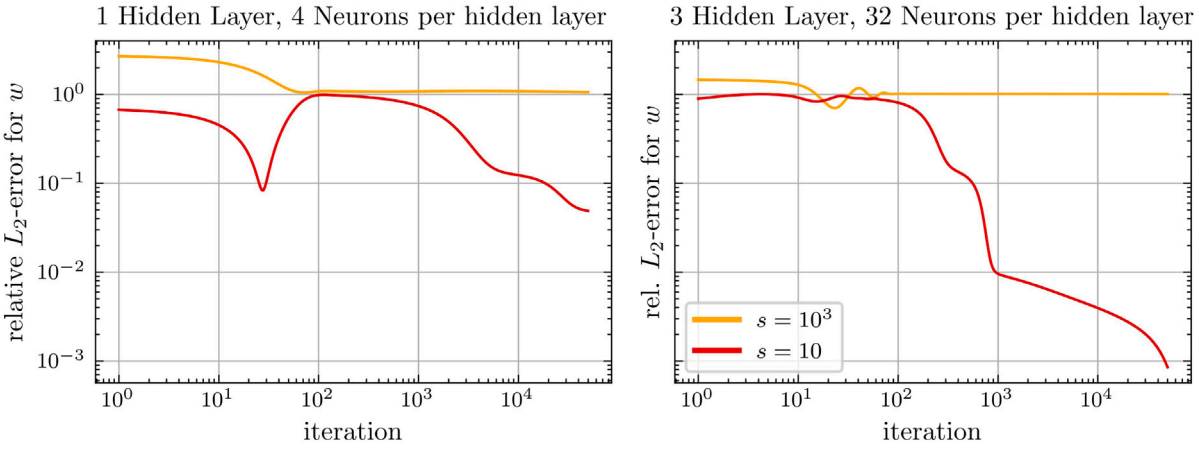


Fig. A.27. Evolution of the relative L_2-error of the displacement for a small network (left) and a large network (right), each evaluated on a thick and a slender beam.

Comparing the activation functions, GELU tends to yield lower errors than Tanh, particularly for the stress resultants. This may be related to the smoother and stronger gradient propagation of GELU.

To assess the influence of network size on shear locking, we compare a small and a large architecture based on the baseline (w, φ)-parametrization and track the relative L_2-errors during training for a thick and a slender beam. The results are shown in Fig. A.27. For the thick beam, the larger network accelerates training. For the slender beam, however, no meaningful improvement is observed for the displacement w: shear locking occurs for both architectures, supporting the interpretation that insufficient network capacity is not the sole cause of the observed locking behavior.

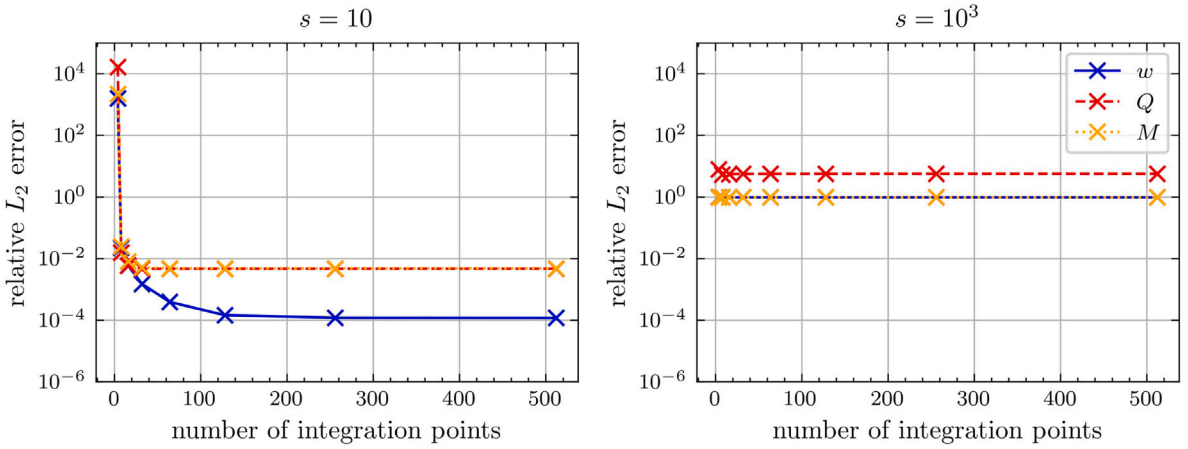


Fig. B.28. Final relative L_2-errors as a function of the number of integration points for the baseline (w, φ)-parametrization for a thick beam (left) and a slender beam (right).

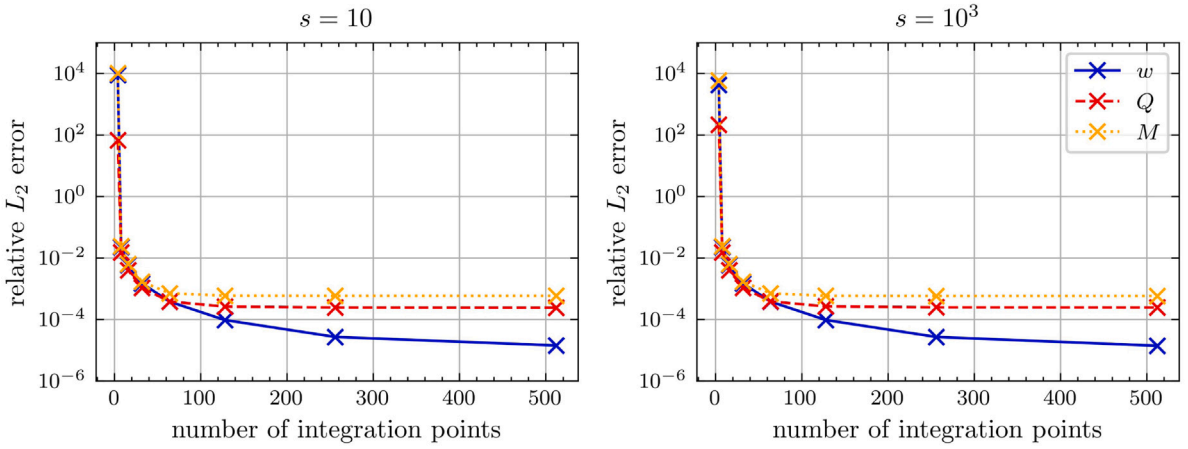


Fig. B.29. Final relative L_2-errors as a function of the number of integration points for the modified (w, γ)-parametrization for a thick beam (left) and a slender beam (right).

Based on this study, all PINNs using two neural networks employ two hidden layers with 32 neurons per layer and the GELU activation function, resulting in 4418 trainable parameters. For a fair comparison, PINNs based on a single neural network use 46 neurons per hidden layer, which yields a comparable number of trainable parameters, namely 4510.

Appendix B. Quadrature study

The required integrals are evaluated by Gauss quadrature. To determine a suitable number of integration points, we solve the beam example from Section 6.1 for both a thick and a slender beam while varying the number of quadrature points from 4 to 512. Each PINN is trained for 50,000 iterations. The final errors for the baseline (w, φ)-PINN are shown in Fig. B.28, and the corresponding results for the locking-free (w, γ)-formulation are shown in Fig. B.29.

For the baseline PINN, locking persists across the full range of quadrature points, showing that insufficient quadrature is not the cause of the observed locking behavior. For both formulations, the errors in the stress resultants reach a plateau once more than approximately 128 integration points are used, indicating that further reduction of the quadrature error does not lead to a meaningful improvement. Based on these results, all numerical examples in this work use 128 quadrature points, unless reported otherwise.

Data availability

Data will be made available on request.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[2] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.

[3] E. Kharazmi, Z. Zhang, G.E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, 2019, http://dx.doi.org/10.48550/arXiv.1912.00873, arXiv:1912.00873.

[4] W. E, B. Yu, The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12, http://dx.doi.org/10.1007/s40304-018-0127-z.

[5] V.M. Nguyen-Thanh, X. Zhuang, T. Rabczuk, A deep energy method for finite deformation hyperelasticity, Eur. J. Mech. A Solids 80 (2020) 103874, http://dx.doi.org/10.1016/j.euromechsol.2019.103874.

[6] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741, http://dx.doi.org/10.1016/j.cma.2021.113741.

[7] N. Radin, S. Klinkel, O. Altay, Effects of variational formulations on physics-informed neural network performance in solid mechanics, PAMM 23 (4) (2023) e202300222, http://dx.doi.org/10.1002/pamm.202300222.

[8] L. Wang, G. Liu, G. Wang, K. Zhang, M-PINN A mesh-based physics-informed neural network for linear elastic problems in solid mechanics, Internat. J. Numer. Methods Engrg. 125 (9) (2024) e7444, http://dx.doi.org/10.1002/nme.7444.

[9] D.W. Abueidda, S. Koric, R.A. Al-Rub, C.M. Parrott, K.A. James, N.A. Sobh, A deep learning energy method for hyperelasticity and viscoelasticity, Eur. J. Mech. A Solids 95 (2022) 104639, http://dx.doi.org/10.1016/j.euromechsol.2022.104639.

[10] S.B. Tandale, F. Bamer, B. Markert, M. Stoffel, Physics-based self-learning recurrent neural network enhanced time integration scheme for computing viscoplastic structural finite element response, Comput. Methods Appl. Mech. Engrg. 401 (2022) 115668, http://dx.doi.org/10.1016/j.cma.2022.115668.

[11] W. Li, M.Z. Bazant, J. Zhu, A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches, Comput. Methods Appl. Mech. Engrg. 383 (2021) 113933, http://dx.doi.org/10.1016/j.cma.2021.113933.

[12] X. Zhuang, H. Guo, N. Alajlan, H. Zhu, T. Rabczuk, Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning, Eur. J. Mech. A Solids 87 (2021) 104225, http://dx.doi.org/10.1016/j.euromechsol.2021.104225.

[13] J.-H. Bastek, D.M. Kochmann, Physics-informed neural networks for shell structures, Eur. J. Mech. A Solids 97 (2023) 104849, http://dx.doi.org/10.1016/j.euromechsol.2022.104849.

[14] T. Sahin, M. von Danwitz, A. Popp, Solving forward and inverse problems of contact mechanics using physics-informed neural networks, Adv. Model. Simul. Eng. Sci. 11 (1) (2024) 11, http://dx.doi.org/10.1186/s40323-024-00265-3.

[15] Z. Li, J. Bai, H. Ouyang, S. Martelli, M. Tang, Y. Yang, H. Wei, P. Liu, R. Wei, Y. Gu, Physics-informed neural networks for friction-involved nonsmooth dynamics problems, Nonlinear Dynam. 112 (9) (2024) 7159–7183, http://dx.doi.org/10.1007/s11071-024-09350-z.

[16] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theor. Appl. Fract. Mech. 106 (2020) 102447, http://dx.doi.org/10.1016/j.tafmec.2019.102447.

[17] Y. Ghaffari Motlagh, P.K. Jimack, R. de Borst, Deep learning phase-field model for brittle fractures, Internat. J. Numer. Methods Engrg. 124 (3) (2023) 620–638, http://dx.doi.org/10.1002/nme.7135.

[18] B. Zheng, T. Li, H. Qi, L. Gao, X. Liu, L. Yuan, Physics-informed machine learning model for computational fracture of quasi-brittle materials without labelled data, Int. J. Mech. Sci. 223 (2022) 107282, http://dx.doi.org/10.1016/j.ijmecsci.2022.107282.

[19] J. He, C. Chadha, S. Kushwaha, S. Koric, D. Abueidda, I. Jasiuk, Deep energy method in topology optimization applications, Acta Mech. 234 (4) (2023) 1365–1379, http://dx.doi.org/10.1007/s00707-022-03449-3.

[20] H. Jeong, J. Bai, C. Batuwatta-Gamage, C. Rathnayaka, Y. Zhou, Y. Gu, A physics-informed neural network-based topology optimization (PINNTO) framework for structural optimization, Eng. Struct. 278 (2023) 115484, http://dx.doi.org/10.1016/j.engstruct.2022.115484.

[21] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, Sci. Adv. 7 (40) (2021) eabi8605, http://dx.doi.org/10.1126/sciadv.abi8605.

[22] S. Goswami, M. Yin, Y. Yu, G.E. Karniadakis, A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials, Comput. Methods Appl. Mech. Engrg. 391 (2022) 114587, http://dx.doi.org/10.1016/j.cma.2022.114587.

[23] M.S. Eshaghi, C. Anitescu, M. Thombre, Y. Wang, X. Zhuang, T. Rabczuk, Variational physics-informed neural operator (VINO) for solving partial differential equations, 2024, https://arxiv.org/abs/2411.06587v1.

[24] W. Cho, M. Jo, H. Lim, K. Lee, D. Lee, S. Hong, N. Park, Parameterized physics-informed neural networks for parameterized PDEs, 2024, http://dx.doi.org/10.48550/arXiv.2408.09446, arXiv:2408.09446.

[25] D. Chapelle, K.-J. Bathe, The Finite Element Analysis of Shells - Fundamentals, in: Computational Fluid and Solid Mechanics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, http://dx.doi.org/10.1007/978-3-642-16408-8.

[26] M. Bischoff, Finite elements for plates and shells, in: Encyclopedia of Continuum Mechanics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2020, pp. 1–23, http://dx.doi.org/10.1007/978-3-662-55771-6.

[27] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39) (2005) 4135–4195, http://dx.doi.org/10.1016/j.cma.2004.10.008.

[28] T. Elguedj, Y. Bazilevs, V.M. Calo, T.J.R. Hughes, B-bar and F-bar projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements, Comput. Methods Appl. Mech. Engrg. 197 (33) (2008) 2732–2762, http://dx.doi.org/10.1016/j.cma.2008.01.012.

[29] R. Echter, M. Bischoff, Numerical efficiency, locking and unlocking of NURBS finite elements, Computational Geometry and Analysis, Comput. Methods Appl. Mech. Engrg. 199 (5) (2010) 374–382, http://dx.doi.org/10.1016/j.cma.2009.02.035.

[30] Q. Long, P. Burkhard Bornemann, F. Cirak, Shear-flexible subdivision shells, Internat. J. Numer. Methods Engrg. 90 (13) (2012) 1549–1577, http://dx.doi.org/10.1002/nme.3368.

[31] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, P. Krysl, Meshless methods: An overview and recent developments, Comput. Methods Appl. Mech. Engrg. 139 (1) (1996) 3–47, http://dx.doi.org/10.1016/S0045-7825(96)01078-X.

[32] M. Arroyo, M. Ortiz, Local maximum-entropy approximation schemes: A seamless bridge between finite elements and meshfree methods, Internat. J. Numer. Methods Engrg. 65 (13) (2006) 2167–2202, http://dx.doi.org/10.1002/nme.1534.

[33] S. Bieber, B. Oesterle, E. Ramm, M. Bischoff, A variational method to avoid locking—Independent of the discretization scheme, Internat. J. Numer. Methods Engrg. 114 (8) (2018) 801–827, http://dx.doi.org/10.1002/nme.5766.

[34] L. Beirão da Veiga, C. Lovadina, A. Reali, Avoiding shear locking for the Timoshenko beam problem via isogeometric collocation methods, Comput. Methods Appl. Mech. Engrg. 241–244 (2012) 38–51, http://dx.doi.org/10.1016/j.cma.2012.05.020.

[35] E. Marino, Isogeometric collocation for three-dimensional geometrically exact shear-deformable beams, Comput. Methods Appl. Mech. Engrg. 307 (2016) 383–410, http://dx.doi.org/10.1016/j.cma.2016.04.016.

[36] J. Kiendl, F. Auricchio, L. Beirão da Veiga, C. Lovadina, A. Reali, Isogeometric collocation methods for the Reissner–Mindlin plate problem, Isogeometric Analysis Special Issue, Comput. Methods Appl. Mech. Engrg. 284 (2015) 489–507, http://dx.doi.org/10.1016/j.cma.2014.09.011.

[37] J. Kiendl, E. Marino, L. De Lorenzis, Isogeometric collocation for the Reissner–Mindlin shell problem, Comput. Methods Appl. Mech. Engrg. 325 (2017) 645–665, http://dx.doi.org/10.1016/j.cma.2017.07.023.
[38] J. Dick, S. Ko, Q.T.L. Gia, K. Mustapha, S. Park, A decomposition-based robust training of physics-informed neural networks for nearly incompressible linear elasticity, 2025, http://dx.doi.org/10.48550/arXiv.2505.21994, arXiv:2505.21994.
[39] B. Oesterle, E. Ramm, M. Bischoff, A shear deformable, rotation-free isogeometric shell formulation, Comput. Methods Appl. Mech. Engrg. 307 (2016) 235–255, http://dx.doi.org/10.1016/j.cma.2016.04.015.
[40] B. Oesterle, S. Bieber, R. Sachse, E. Ramm, M. Bischoff, Intrinsically locking-free formulations for isogeometric beam, plate and shell analysis, PAMM 18 (1) (2018) e201800399, http://dx.doi.org/10.1002/pamm.201800399.
[41] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: A survey, pp. 43.
[42] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, Comput. Methods Appl. Mech. Engrg. 362 (2020) 112790, http://dx.doi.org/10.1016/j.cma.2019.112790.
[43] J. Müller, M. Zeinhofer, Achieving high accuracy with PINNs via energy natural gradient descent, in: Proceedings of the 40th International Conference on Machine Learning, PMLR, 2023, pp. 25471–25485.
[44] S. Berrone, C. Canuto, M. Pintore, N. Sukumar, Enforcing Dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks, Heliyon 9 (8) (2023) e18820, http://dx.doi.org/10.1016/j.heliyon.2023.e18820.
[45] R. Echter, B. Oesterle, M. Bischoff, A hierarchic family of isogeometric shell finite elements, Comput. Methods Appl. Mech. Engrg. 254 (2013) 170–180, http://dx.doi.org/10.1016/j.cma.2012.10.018.
[46] L. Beirão Da Veiga, T.J.R. Hughes, J. Kiendl, C. Lovadina, J. Niiranen, A. Reali, H. Speleers, A locking-free model for Reissner–Mindlin plates: Analysis and isogeometric implementation via NURBS and triangular NURPS, Math. Models Methods Appl. Sci. 25 (08) (2015) 1519–1551, http://dx.doi.org/10.1142/S0218202515500402.
[47] B. Oesterle, R. Sachse, E. Ramm, M. Bischoff, Hierarchic isogeometric large rotation shell elements including linearized transverse shear parametrization, Comput. Methods Appl. Mech. Engrg. 321 (2017) 383–405, http://dx.doi.org/10.1016/j.cma.2017.03.031.
[48] R.A. Anderson, Transient Respone of Uniform Beams (Ph.D. thesis), California Institue of Technology, Pasadena, California, 1953.
[49] K. Marguerre, H. Wölfel, Mechanics of Vibration, Sijthoff & Noordhoff, Alphen aan den Rijn, 1979.
[50] Z. Chen, V. Badrinarayanan, C.-Y. Lee, A. Rabinovich, GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks.
[51] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, J. Comput. Phys. 449 (2022) 110768, http://dx.doi.org/10.1016/j.jcp.2021.110768.
[52] R. Bischof, M.A. Kraus, Multi-objective loss balancing for physics-informed deep learning, Comput. Methods Appl. Mech. Engrg. 439 (2025) 117914, http://dx.doi.org/10.1016/j.cma.2025.117914.
[53] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366, http://dx.doi.org/10.1016/0893-6080(89)90020-8.
[54] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (6) (1993) 861–867, http://dx.doi.org/10.1016/S0893-6080(05)80131-5.
[55] D. Hendrycks, K. Gimpel, Gaussian error linear units (GELUs), 2023, http://dx.doi.org/10.48550/arXiv.1606.08415, arXiv:1606.08415.
[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, 2019, http://dx.doi.org/10.48550/arXiv.1912.01703, arXiv:1912.01703.
[57] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, arXiv:1412.6980.
[58] S.J. Reddi, S. Kale, S. Kumar, On the convergence of adam and beyond, in: International Conference on Learning Representations, 2018.
[59] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, 2015, http://dx.doi.org/10.48550/arXiv.1502.01852, arXiv:1502.01852.
[60] T. Le-Duc, T.N. Vo, H. Nguyen-Xuan, J. Lee, On the Gauss–Legendre quadrature rule of deep energy method for one-dimensional problems in solid mechanics, Finite Elem. Anal. Des. 242 (2024) 104248, http://dx.doi.org/10.1016/j.finel.2024.104248.
[61] X. Wang, J. Zhao, Z.-Y. Yin, X. Zhuang, Failure mechanisms and resolution in deep energy method, Int. J. Mech. Sci. 313 (2026) 111278, http://dx.doi.org/10.1016/j.ijmecsci.2026.111278.
[62] C. Chinwuba Ike, Mathematical solutions for the flexural analysis of Mindlin's first order shear deformable circular plates, Math. Model. Eng. 4 (2) (2018) 50–72, http://dx.doi.org/10.21595/mme.2018.19825.
[63] C. Chinosi, C. Lovadina, Numerical analysis of some mixed finite element methods for Reissner-Mindlin plates, Comput. Mech. 16 (1) (1995) 36–44, http://dx.doi.org/10.1007/BF00369883.